VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Implementace herních strategií pro hru HEX

Implementation of Game Strategies for HEX Game

2018 Jan Polok

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jan Polok**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Implementace herních strategií pro hru HEX
Implementation of Game Strategies for HEX Game

Jazyk vypracování: čeština

Zásady pro vypracování:

Hlavním cílem práce je implementace herních strategií pro hru HEX, včetně prostředí s uživatelským rozhraním pro simulaci této hry. HEX je strategická desková hra, kterou je možno analyzovat s využitím pojmů z teorie grafů. Rozhraní bude umožňovat hru proti počítači – umělé inteligenci, kde budou využity implementované strategie.

Práce bude obsahovat:
1. Seznámení se hrou HEX.
2. Přehled volně dostupných implementací hry HEX s umělou inteligencí.
3. Implementace hry a herních strategií.
4. Porovnání vlastní implementace s implementacemi uvedenými v bodu 2.

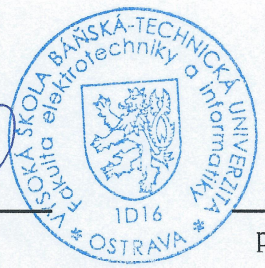
Seznam doporučené odborné literatury:

[1] BROWNE, Cameron. Hex strategy: making the right connections. Natick, Mass: A.K. Peters, c2000. ISBN 978-1568811178.
[2] P. Kovář, Teorie grafů, učební text. Dostupné také z:
http://homel.vsb.cz/~kov16/files/skriptum_teorie_grafu_rozsirene.pdf

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Michal Fait**

Datum zadání: 01.09.2017
Datum odevzdání: 13.07.2018



doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 12. července 2018 .....

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli, zejména Mgr. Daniele Kolářové Jasenské za pomoc při shánění literárních zdrojů a Ing. Michalovi Faitovi za vedení a cenné rady při tvorbě práce.

Abstrakt

Bakalářská práce se věnuje seznámení se hrou Hex a jejími herními strategiemi. Cílem práce je vytvoření vlastní implementace hry včetně prostředí, které umožňuje simulaci hry. Prostředí musí umožňovat různé herní možnosti včetně hry dvou hráčů a hry hráče proti počítači. Ve vlastní implementaci musí být použity různé herní strategie, které bude využívat počítač při hře proti uživateli. Práce se dále zabývá přehledem volně dostupných implementací hry s umělou inteligencí a porovnání těchto implementací s vlastní implementací.

Klíčová slova: Hra Hex, herní strategie, implementace herních strategií, umělá inteligence

Abstract

This bachelor thesis deals with getting to know the Hex game and its strategies. The aim of the work is to create custom implementation of the game including simulation enviroment. The enviroment has to be capable of simulating various game options including player vs. player mode and player vs. PC mode. The custom implementation has to use various game strategies, which would be used by the PC during player vs. PC mode. This thesis also maps freely available game implementations with AI and compares those implementations with the new, custom one.

Key Words: Hex game, game strategies, implementation of game strategies, artificial intelligence

Obsah

Seznam použitých zkratek a symbolů

AI	–	Artificial Intelligence
BFS	–	Breadth-First Search
CPU	–	Central Processing Unit
DFS	–	Depth-First Search
GUI	–	Graphical User Interface
OS	–	Operating System
SBFS	–	Spanning-BFS
SemiSBFS	–	Semi-Spanning-Breadth-First Search

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Hry a herní průmysl jsou nedílnou součástí dnešního světa. Některé slouží pouze k pobavení, ale mnoho her slouží také ke vzdělávání. Existuje mnoho her, které jsou celosvětově známé a proslulé. Šachy jsou skvělým příkladem. Jejich hra je v Evropě známá přes pět století a dodnes se těší velké oblibě. Ať už se jedná o jakoukoli hru, touhou všech hráčů je vyhrát. Co když je možné nespoléhat na instinkt a štěstí, ale pokusit se logickým myšlením dojít k postupu, jak vyhrát? Je možné tento postup opakovat? Tyto otázky vedly ke snaze analyzovat různé hry a popsat doporučené herní postupy, které jsou opakovaně aplikovatelné.

Dnes víme, že existují herní strategie pro každou variantu jakékoli hry. Začátečník může znát pouze omezené množství herních taktik, což mu znemožní šance na výhru proti expertnímu hráči. Na druhou stranu, i dlouholetý ostřílený hráč může přehlédnout základní chybu, protože přemýšlí nad složitou analýzou a nepředpokládá začátečnické tahy.

Od prvopočátků počítačů a výpočetní techniky zde byly pokusy naučit počítač samostatně myslet. Jedná se o tzv. umělou inteligenci (artificial intelligence), kdy je počítač schopen analyzovat nově vzniklou situaci a vhodným způsobem na ni reagovat. Vývojáři implementují AI do her, aby počítač mohl zastávat pozici konkurenceschopného hráče.

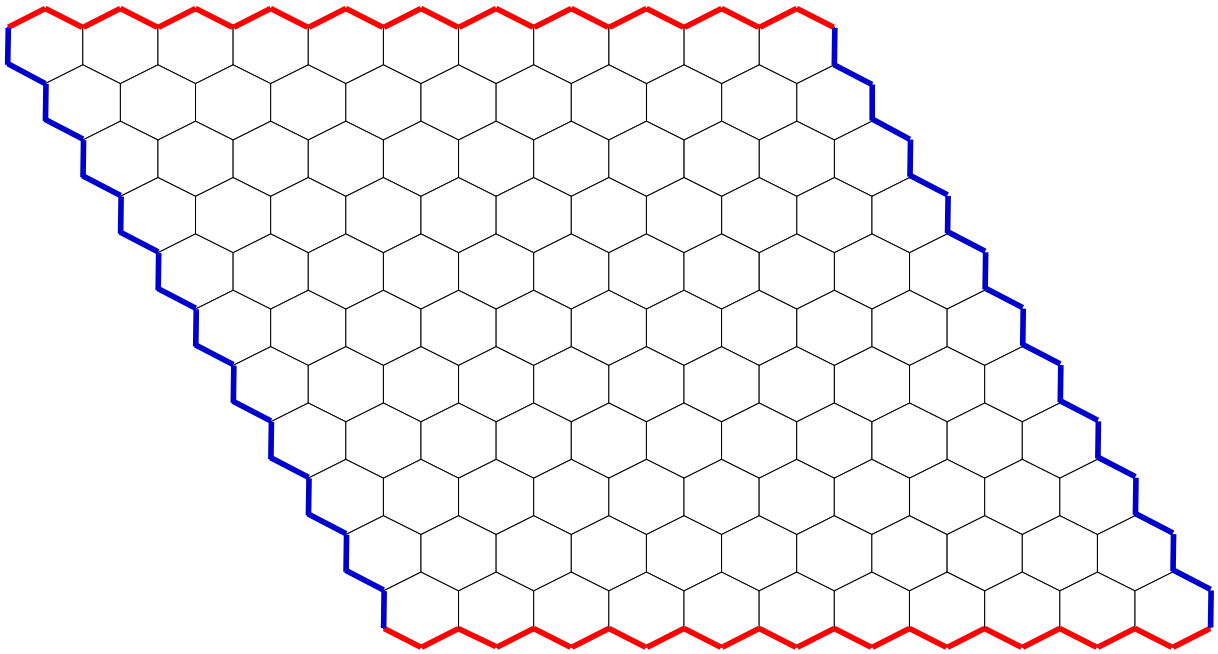
Tato práce se zabývá analýzou hry Hex. První kapitoly popisují pravidla, základní pojmy, struktury i strategie a popisují jejich výhody a nevýhody. Dále je představen program, který byl vytvořen v rámci praktické části práce - vlastní implementace hry Hex s umělou inteligencí a testovacím prostředím. Poslední část této práce mapuje volně dostupné implementace hry Hex s umělou inteligencí a zabývá se testováním a porovnáváním vlastního programu s ostatními programy.

2 Seznámení se hrou Hex

Tato kapitola popisuje hru Hex z historického hlediska, její pravidla a varianty.

2.1 Popis hry

Hex je logická desková hra pro dva hráče. Cílem hry je vytvořit souvislý řetěz z 'kamenů' své barvy, který spojuje dvě stejně barevné protilehlé strany herní desky. Hráč, který takovýto řetěz dokáže vystavět, vítězí. [11] Herní deska je tvarována do kosočtverce. V tomto kosočtverci se nachází určitý počet herních pozic (buněk), které jsou tvarované do šestiúhelníků (hexagonů). Počet hexagonů závisí na velikosti herní desky, která má rozměr $n * n$ hexagonů. Nejčastější velikost herní desky je $11 * 11$. Obrázek č. 1 zobrazuje prázdnou herní desku o velikosti 11. Herní desku je možno jakkoli natočit. Tato práce i vlastní program operuje pouze s tímto natočením herní desky. Horní a spodní strana jsou červené strany, které má za úkol propojit červený hráč, zatímco levá a pravá strana jsou strany modrého hráče. Šestiúhelník, který se nachází ve kterémkoli rohu herní desky se nazývá tzv. *rohový hexagon* a sousedí zároveň s modrou i červenou stranou herní desky.



Obrázek 1: Prázdná herní deska

Desková varianta hry obsahuje herní desku a kameny dvou barev. Položením kamenu na hexagon daný hráč obsadí tento hexagon. V elektronické variantě hry už kameny nejsou potřeba, stačí daný hexagon přebarvit na barvu hráče.

V této práci i vlastním programu se bílý hexagon považuje za neobsazený (volný). Kamenem se v této práci rozumí modrý nebo červený hexagon na desce. Pokud je v této práci obecně

zmíněn hexagon bez bližšího popisu, jedná se o buňku na herní desce libovolné barvy. Termín položení kamenu na hexagon je v této práci považován za ekvivalentní termínu označit hexagon danou barvou - jde o stejnou akci obsazení bílého hexagonu jedním z hráčů.

Partie Hex nemůže nikdy skončit remízou. [12, s. 73] I když jsou všechny hexagony obsazené, na herní desce se bude vždy nakonec nacházet alespoň jeden vítězný řetěz kamenů. Hex je proto tzv. deterministická hra [4, s. 7]. Hex se také řadí mezi abstraktní hry. Takové hry nepotřebují úvod do příběhu, stačí znát pravidla.

2.2 Pravidla hry

Pravidla hry Hex jsou jedny z nejjednodušších. I když je snadné jim porozumět, umět dobře hrát Hex je složité. Pravidla jsou shrnutelné do několika bodů. [4, s. 2]

- Na začátku hry jsou všechny hexagony na desce neobsazené.
- Partii začíná červený hráč.
- Každý hráč ve svém tahu položí právě jeden kámen své barvy na jakýkoli volný hexagon.
- Hráči se po položení kamenu střídají.
- Položené kameny jsou neměnné, není je možné brát, či vyměnit (jedinou výjimkou je swapovací pravidlo, viz kapitola 2.2.1).
- Hra končí, když jeden z hráčů propojí souvislým řetězem kamenů obě protilehlé strany své barvy.

Červený hráč, který hru začíná, má velkou výhodu položení prvního kamenu na desce. Na desce o velikosti 7 a menší existují vítězné strategie, kdy stačí správně umístit první kámen a vítězství červeného hráče je zaručeno (za předpokladu, že v příštích tazích neudělá chybu)[2, s. 2]. Pro vyvážení šancí na výhru mezi hráči se často přidává tzv. *Swap Rule* [4, s. 2], česky swapovací pravidlo. Před začátkem každé partie je nutné se dohodnout, zda se s tímto pravidlem hraje.

2.2.1 Swap Rule

Swapovací pravidlo znemožňuje červenému hráči udělat příliš silný první tah, který by mu zaručil, nebo zvýšil šance na výhru. Swapovací pravidlo lehce pozměňuje obecné pravidla pouze pro první tah modrého hráče.

- Modrý hráč může ve svém prvním tahu místo položení kamene na neobsazenou pozici vyměnit již položený červený kámen za modrý.

Tohle je jediná možnost, kdy se na herní desce může manipulovat s položeným kamenem. Pokud dojde ke swapu, červený hráč na začátku svého druhého tahu nemá na desce položený žádný kámen a ztratil tak výhodu prvního položeného kamenu [14, s. 46]. Červený hráč je za použití tohoto pravidla nucen provést slabší první tah, který mu nezaručí či nezvýší šanci na výhru.

2.3 Historie

První publikovaný popis hry z roku 1942 se přisuzuje dánskému matematikovi jménem Piet Hein. Hein ve svém článku popsal hru jménem Polygon, nicméně deska a princip hry je totožný se hrou Hex. Tento dánský matematik je považován za autora hry. Nezávisle na Heinovi byl Hex vytvořen americkým matematikem jménem John Nash v roce 1948. V roce 1968 byla vydána desková hra pod názvem *Con-Tac-Tix* s deskou o velikosti 12. Od 50. let probíhají pokusy o implementaci hry na různých výpočetních zařízeních. Na přelomu tisíciletí byla hra velmi populární a Hex se zařadil i jako disciplína pro počítačové olympiády. Na těchto olympiádách soutěží různé implementace umělých inteligencí her mezi sebou. [1, s. 2]

2.4 Jiné varianty hry

Výše popsaná varianta je základní a nejrozšířenější varianta hry Hex. Veškeré implementace, algoritmy a postupy se v této práci budou vztahovat k této základní variantě hry. Postupem času se vyvinuly další varianty a alternativy, které jsou popsané v této sekci.

2.4.1 Hra Y

Hra Y je odlišná od hry Hex tím, že se hraje na herní desce ve tvaru trojúhelníku. Jelikož herní plán má pouze tři strany, cílem obou hráčů je vytvořit řetěz z kamenů dotýkající se všech tří stran. Tento prinicip adaptovala např. vědomostní soutěž AZ-kvíz od České televize [8].

2.4.2 Hra do čtverce

Hra do čtverce se od Hex liší tím, že se hraje na čtvercové herní desce se čtvercovými políčky namísto šestiúhelníků. Tato varianta hry může skončit remízou. Ani plně zaplněná herní deska negarantuje spojení vítězného řetězu. Když už je jasné, že hru nemůže vyhrát ani jeden hráč, nastává tzv. *deadlock* a nemá ve hře smysl pokračovat[4, s. 34].

2.4.3 Hex pro tři hráče

Hex je hratelný i ve větším počtu hráčů. Varianta pro tři hráče se hraje na desce o tvaru šestiúhelníku, kde má každý hráč za úkol spojit dvě protilehlé strany. Tento typ se nazývá tzv. *Metahex*. Stejně jako u hry do čtverce může Metahex skončit remízou [4, s. 40].

3 Obecné pojmy, struktury a strategie

Tato kapitola popisuje obecné pojmy, vybrané struktury a strategie hry Hex.

Struktura je určitá formace hexagonů, která je charakteristická svým chováním. Cílem struktur je zjednodušit analýzu herní desky. I když základní jednotkou hry je hexagon, pro analýzu desky se nevyplatí vnímat hexagony samostatně. Vhodnější přístup je vnímání více hexagonů jako větší celky, se kterými je jednodušší dále pracovat. Struktury jsou obecné a aplikovatelné pro oba hráče.

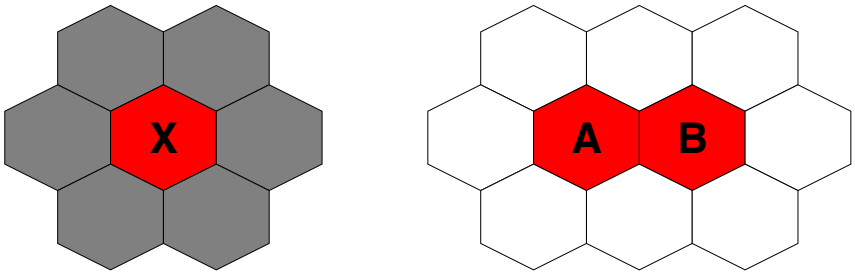
Strategiemi se rozumí obecně doporučené herní taktiky, podle kterých se vyplatí v určitých situacích postupovat.

3.1 Obecné pojmy a struktury

Tato kapitola popisuje základní pojmy hry Hex a představuje vybrané hexagonové struktury.

3.1.1 Sousední hexagony

Dva hexagony se nazývají sousední, pokud spolu sdílí hranu. [4, s. 25] Každý hexagon může mít maximálně šest sousedních hexagonů. V levé části obrázku č. 2 je šedou barvou vyznačeno všech šest sousedních hexagonů červeného hexagonu *X*. Pokud jsou dva kameny sousední, tak se navzájem *dotýkají*. V pravé části tohoto obrázku je možno vidět navzájem se dotýkající kameny *A* a *B*.



Obrázek 2: Sousední hexagony

3.1.2 Řetěz

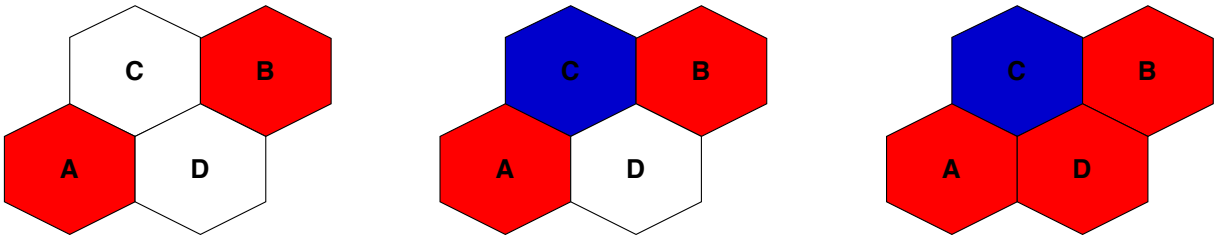
Řetěz je maximální množina kamenů stejné barvy, ve které jsou všechny kameny spolu souvisle propojené pouze pomocí sousedních kamenů. Dva kameny nacházející se ve stejném řetězu se nazývají spojené. [4, s. 27]

3.1.3 Most

Most je základní struktura hry Hex. Most se skládá ze dvou kamenů stejné barvy, které mají dva společné sousední hexagony. Pokud jsou oba tyto společné sousední hexagony bílé, tak mezi

kameny mostu existují dva způsoby, kterými lze vytvořit řetěz obsahující kameny tohoto mostu. [4, s. 29]

Levá část obrázku č. 3 zobrazuje neohrožený most červeného hráče tvořený kameny A a B s bílými společnými sousedními hexagony C a D. Prostřední část tohoto obrázku ilustruje situaci, kdy modrý hráč napadnul tento most položením kamene na hexagon C a tímto tahem zablokoval jednu z možností propojení červených kamenů v tomto mostu. Pokud chce červený hráč zajistit spojení svých kamenů, musí ve svém tahu zachránit tento most položením kamene na hexagon D a vybudovat tak řetěz obsahující kameny A, B a D (pravá část obrázku).



Obrázek 3: Most

3.1.4 Bezpečné spojení

Dva kameny nebo kámen a strana jsou bezpečně spojeny, když jsou přímo spojeny řetězem, nebo je zaručeno případné vytvoření řetězu nezávisle na tom, kam protivník položí svůj kámen v dalším tahu.

Bezpečné spojení či připojení dvou kamenů je realizovatelné pomocí libovolné kombinace řetězů a neohrožených mostů, v případě spojení kamene a strany také pomocí neohrožených šablon (viz kapitola 3.1.15).

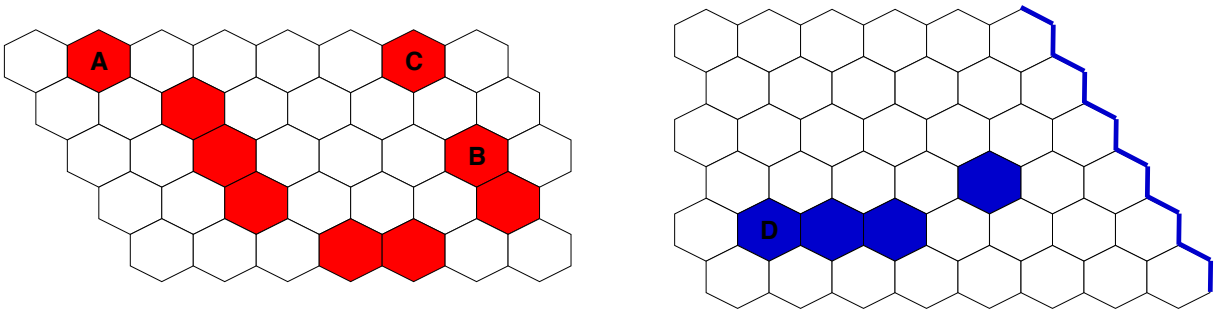
Dva kameny v neohroženém mostu jsou bezpečně spojeny. Neohrožený most se proto také nazývá bezpečně spojený most.

Kameny A a B v levé části obrázku č. 4 jsou bezpečně spojeny za pomocí řetězů a bezpečně spojených mostů. Kámen C není bezpečně spojen s kamenem A. Kámen D v pravé části obrázku je bezpečně připojen k pravé straně desky za pomocí řetězu, mostu a šablony III (viz kapitola 3.1.15).

3.1.5 Průchody

Průchody znázorňují způsob, kterým se dá spojit dva hexagony. Průchody se znázorňují černými čarami.

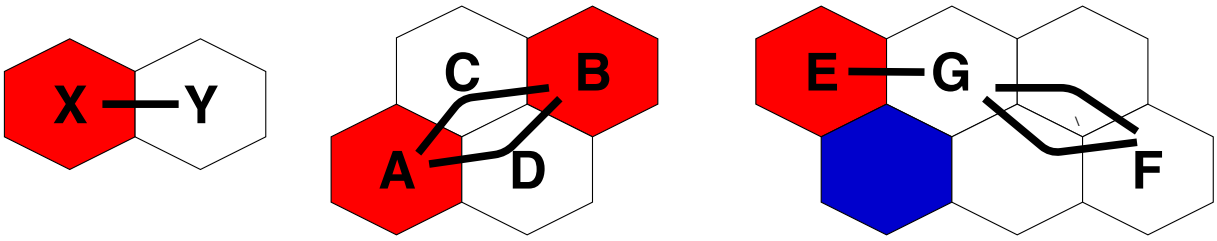
Levá část obrázku 5 zobrazuje přímý průchod z kamene X do bílého hexagonu Y. Tento průchod prochází přes společnou hranu těchto dvou hexagonů. Položením kamene na hexagon Y dojde ke spojení dvou kamenů.



Obrázek 4: Bezpečné spojení

Prostřední část obrázku č. 5 zobrazuje průchody mezi kameny A a B červeného mostu. Oba tyto průchody procházejí přes dvě společné hrany hexagonů A-C a C-B, resp. A-D a D-B. Položením kamene na jeden z bílých hexagonů C nebo D dojde ke spojení kamenů A a B.

Pravá část tohoto obrázku zobrazuje průchody mezi kamenem E a bílým hexagonem F. Jelikož neexistuje žádný přímý průchod z kamene E do F, nelze je v jednom tahu spojit. Pro vytvoření bezpečného spojení mezi E a F je nutno položit kameny na bílé hexagony, ve kterých se nacházejí konce černých čar průchodů, tedy G a F. Pro vytvoření řetězu je pak nutné položit kámen na jeden z bílých hexagonů nacházející se mezi G a F.



Obrázek 5: Průchody

3.1.6 Vzdálenost dvou hexagonů

Vzdálenost dvou hexagonů udává nejmenší počet hexagonů, které je nutné navštívit při procházení přes sousední hexagony od prvního hexagonu ke druhému. Do této vzdálenosti se také započítává cílový hexagon. [4, s. 27]

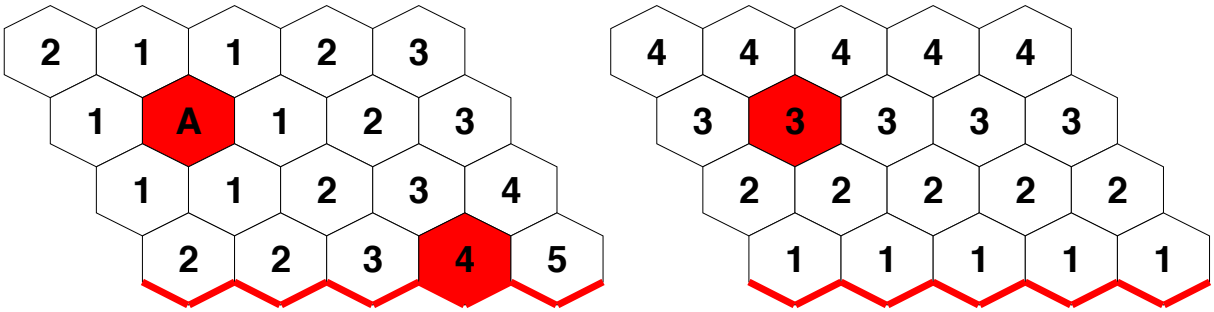
Čísla v hexagonech v levé části obrázku č.6 udávají vzdálenost hexagonů od kamene A. Z této části obrázku je patrné, že červený kámen dotýkající se spodní strany desky se nachází ve vzdálenosti 4 od kamene A.

3.1.7 Vzdálenost hexagonu od strany

Vzdálenost hexagonu od strany udává nejmenší počet hexagonů, které je nutné navštívit při procházení přes sousední hexagony od strany k hexagonu, ke kterému se vzdálenost měří. Do této

vzdálenosti se také započítává cílový hexagon, ke kterému se vzdálenost měří. Pokud se kámen od strany nachází ve vzdálenosti 1, tak se tento kámen a strana *dotýkají*.

Čísla v hexagonech v pravé části obrázku č.6 udávají vzdálenost hexagonů od spodní strany desky. Červený kámen v této části obrázku se nachází ve vzdálenosti 3 od spodní červené strany.



Obrázek 6: Vzdálenost

3.1.8 Konektivita

Konektivita určuje nejmenší počet kamenů, jejichž položení na desku je nezbytné pro vybudování bezpečného spojení mezi dvěma hexagony nebo hexagonem a stranou. Dva kameny nebo kámen a strana, které jsou bezpečně spojené mají mezi sebou konektivitu 0.

Konektivita mostu v levé části obrázku č. 3 je 0, protože kameny A a B jsou bezpečně spojené. Konektivita červených kamenů v prostřední části tohoto obrázku je 1, protože je potřeba položit jeden červený kámen pro obnovení bezpečného spojení. V pravé části popisovaného obrázku je konektivita kamenů A a B opět 0, protože se nacházejí ve stejném řetězu.

Složitější situaci zobrazuje obrázek č. 7, kde konektivita kamenů A a B je 3, protože je nutné položit minimálně tři kameny na šedě označené pozice, které zaručí bezpečné spojení pomocí neohrožených mostů mezi kameny A a B.

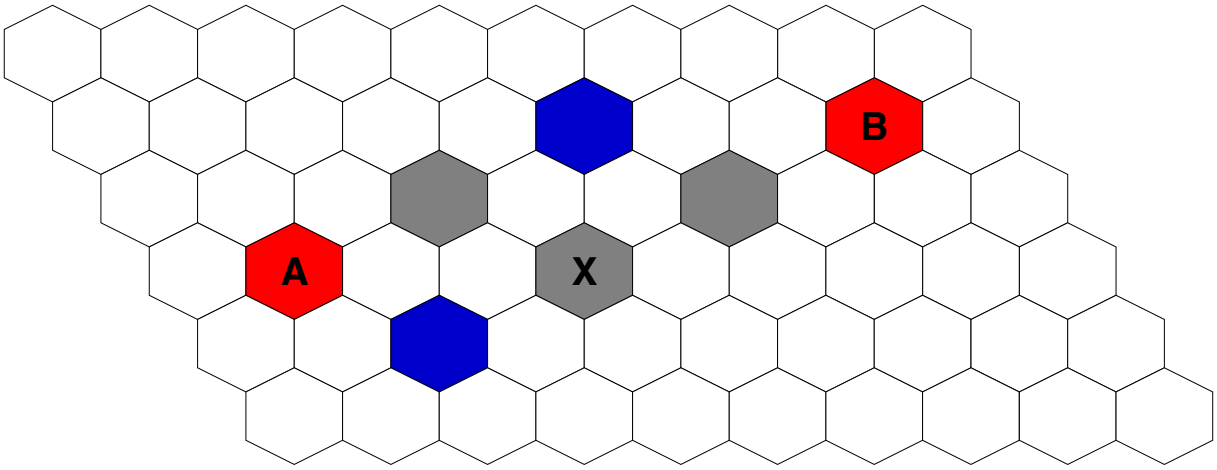
Konektivitu je možno určit i mezi neoznačenými hexagony. Konektivita kamene A a bílého hexagonu X na obrázku č. 7 je 2, protože je potřeba položit dva kameny pro vybudování bezpečného spojení mezi nimi (jeden kámen na pozici X a druhý kámen na šedou pozici mezi A a X).

3.1.9 Krok

Když hráč položí nový kámen X na herní desku a tento kámen je bezpečně připojen ke stejně barevnému kamenu Y, který už na desce leží, provedl hráč krok X z kamenu Y. Krok slouží k vystavění nového bezpečného spojení na herní desce [4, s. 56].

3.1.10 Cesta

V této práci uvažujeme dva druhy cest.



Obrázek 7: Konektivita dvou kamenů na desce

1. Existující cesta je množina již položených kamenů, které bezpečně propojují dva kameny nebo kámen a stranu.

2. Potenciální cesta je množina bílých hexagonů, které je nutno obsadit za účelem vytvoření bezpečného spojení mezi dvěma kameny nebo kamenem a stranou.

Existující cesta mezi dvěma kameny nebo kamenem a stranou se z potenciální cesty buduje pomocí kroků. Konektivita potenciální cesty je určena počtem bílých hexagonů, ze kterých se tato potenciální cesta skládá. Mezi dvěma kameny nebo kamenem a stranou může existovat více potenciálních cest. Nejkratší potenciální cesta je potenciální cesta s nejmenší konektivitiou.

3.1.11 Momentum

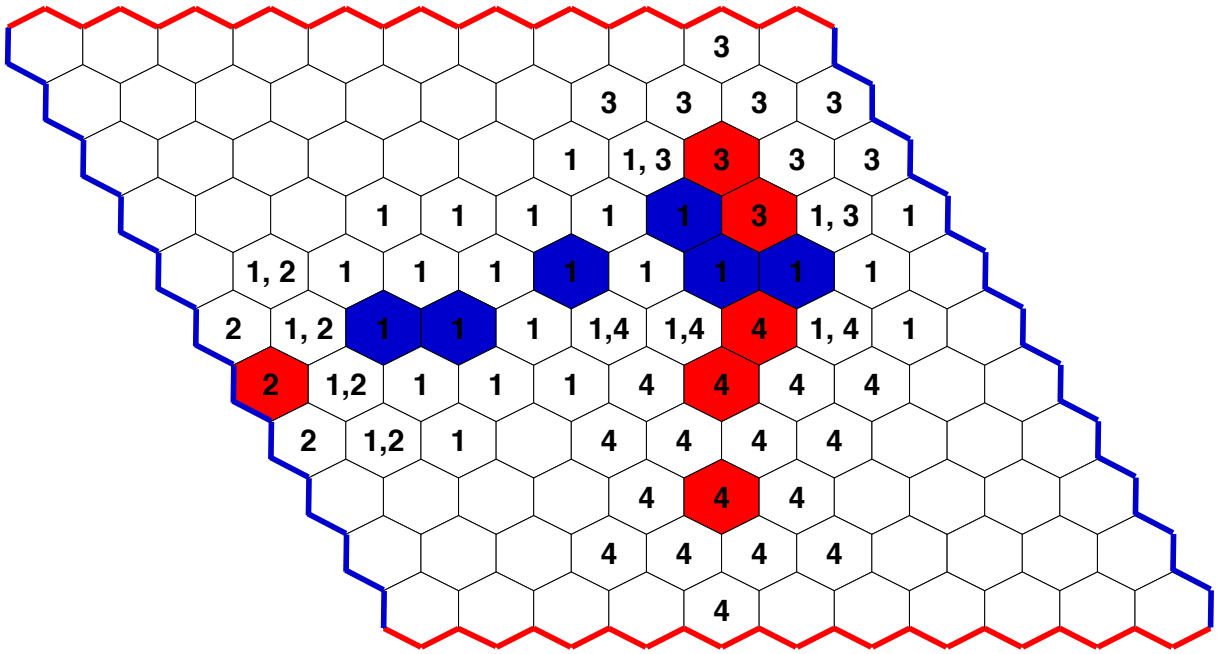
Hráč, který se může rozhodnout, kam ve svém tahu umístí kámen, (jeho kameny nejsou v ohrožení, na které by musel ihned reagovat) se nachází v silné pozici. Takovýto hráč má tzv. *momentum*. Ve vyvážené hře si hráči předávají momentum po každém tahu. Pokud se jeden z hráčů dostane do situace, že musí kontinuálně bránit své struktury nebo blokovat oponenta, ztrácí tento hráč momentum ve prospěch protivníka. Hráč, který má momentum, určuje svými tahy další postup ve hře [4, s. 86].

3.1.12 Skupina

Skupina je struktura, která se skládá ze dvou množin hexagonů. První množina obsahuje bezpečně spojené kameny, zatímco druhá množina skupiny obsahuje bílé hexagony, na které je možno provést krok z kamenů v první množině.

Skupiny slouží k abstrakci a vnímání stejně barevných hexagonů jako větších, bezpečně propojených celků. Obrázek č. 8 popisuje skupiny, které se nacházejí na herní desce. Čísla v hexagonech indikují čísla skupin, ve které se daný hexagon nachází. Pokud je v hexagonu více čísel,

náleží tento hexagon více skupinám. Hexagony bez čísel nenáleží do žádné skupiny. Celkem zde můžeme vidět jednu skupinu modrého hráče a tři skupiny červeného hráče.



Obrázek 8: Skupiny

Definice skupiny se liší v různých zdrojích. Bezpečná skupina popsaná v knize *Hex Strategy: Making the Right Connections* [4, s. 54] chápe množinu bílých hexagonů skupiny jako množinu, ve které jsou obsaženy pouze bílé hexagony, které jsou nezbytnou součástí pro udržení bezpečného spojení kamenů skupiny.

3.1.13 Dvojitý most

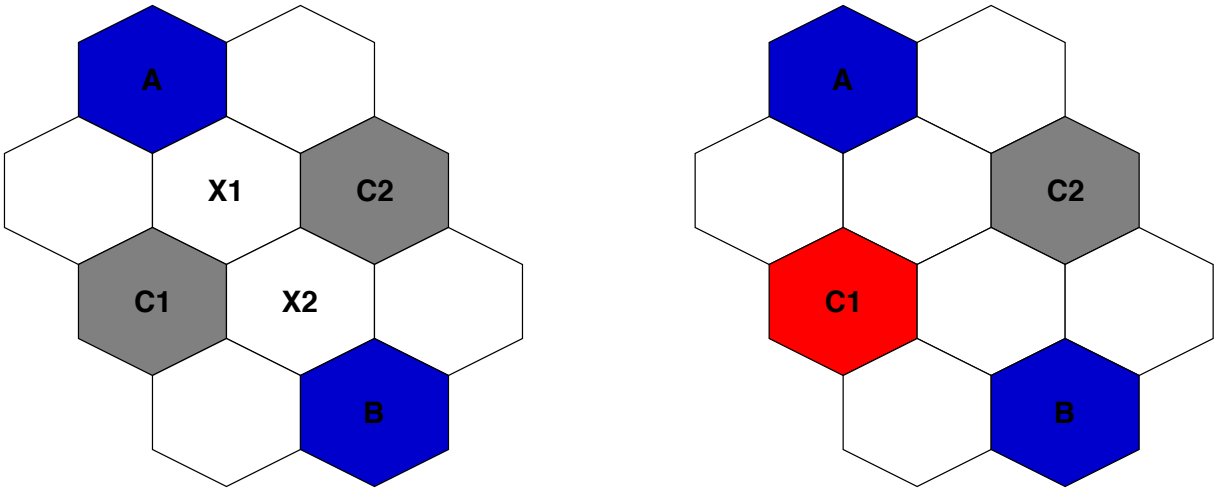
Dvojitý most je struktura, která se skládá ze dvou kamenů stejné barvy, mezi kterými se nachází alespoň jeden bílý hexagon, jehož obsazením by vznikly dva bezpečně spojené mosty. Konektivita dvojitého mostu je 1.

Typický dvojitý most *A-B* modrého hráče prezentuje levá část obrázku č. 9. Modrý hráč dokáže tento dvojitý most snadno přetransformovat do dvou bezpečně spojených mostů položením svého kamene na hexagon *C1* nebo *C2*.

Červený hráč může zablokovat tento dvojitý most položením svého kamene na pozici *X1* nebo *X2*. Pokud však položí svůj kámen na pozici *C1* (pravá část obrázku), modrému hráči stále zbývá volná pozice *C2*, kterou dokáže vybudovat tento dvojitý most.

3.1.14 Volné spojení

Volné spojení je struktura, která se skládá ze dvou kamenů stejné barvy, mezi kterými se nachází alespoň jeden bílý hexagon, jehož obsazením by vznikl jeden bezpečně spojený most s prvním

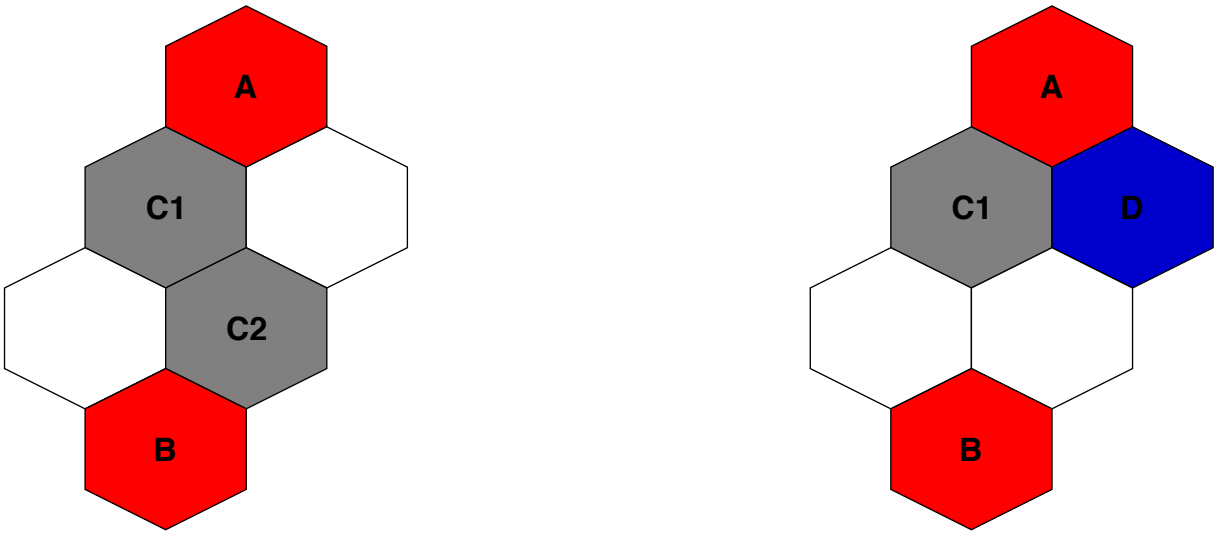


Obrázek 9: Dvojitý most

kamenem a zároveň by se označený hexagon dotýkal druhého kamenu. Konektivita volného spojení je rovna 1, stejně jako u dvojitého mostu.

Typické volné spojení *A*-*B* červeného hráče prezentuje levá část obrázku č. 10. Červený hráč může z tohoto volného spojení vytvořit bezpečně spojený most a dva dotýkající se kameny položením svého kamene na hexagon *C1* nebo *C2*.

Modrý hráč dokáže zablokovat toto volné spojení položením svého kamene na pozici *C1* nebo *C2*. Pokud modrý hráč položí svůj kámen na pozici *D* (pravá část obrázku), zbývá červenému hráči stále jedna možnost, kterou dokáže toto volné spojení zachránit a to položením červeného kamene na pozici *C1*.

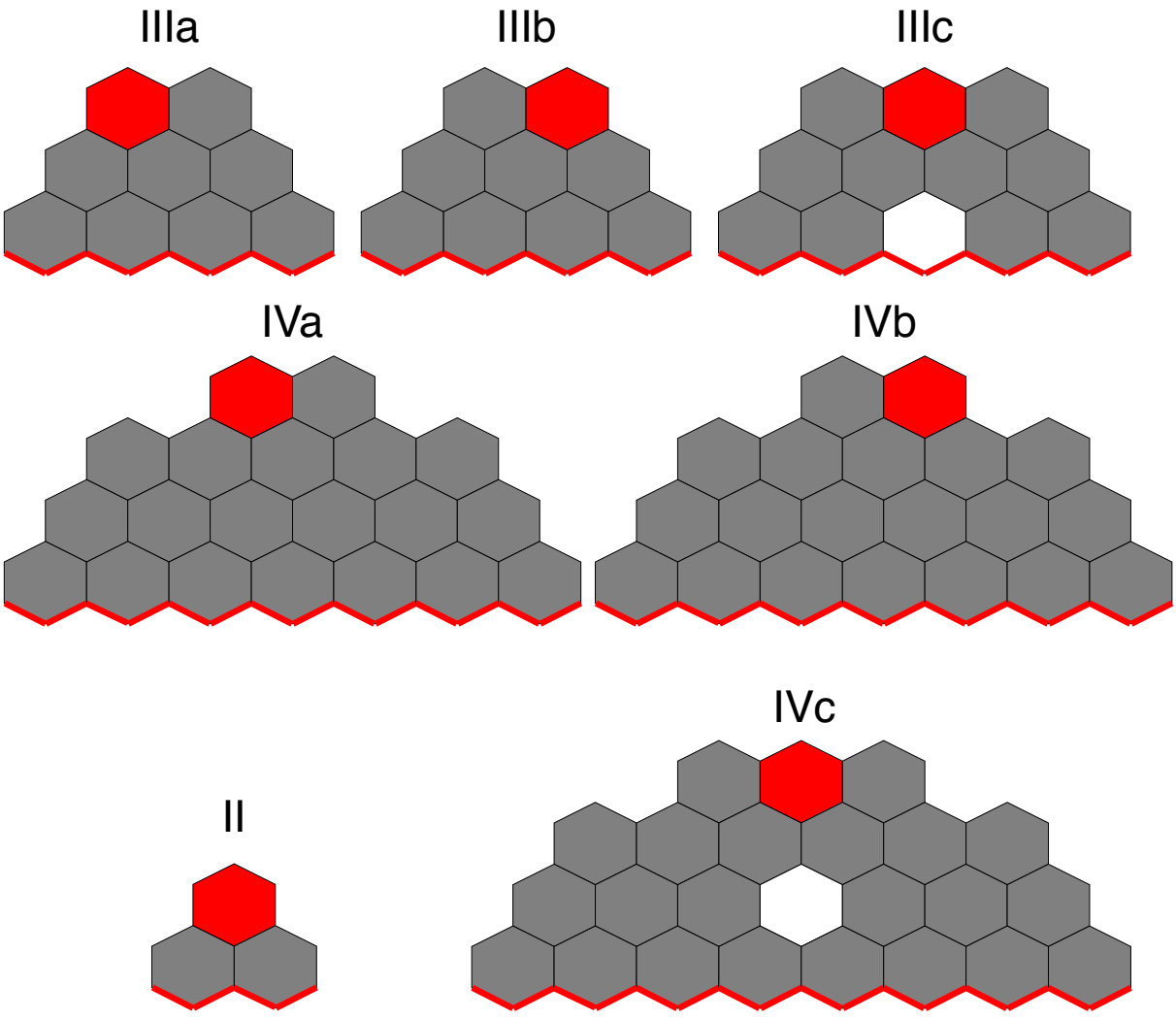


Obrázek 10: Volné spojení

3.1.15 Šablony

Šablony jsou struktury, které se používají k zajištění bezpečného spojení strany herní desky a kamene, který se jí nedotýká. [22] Barevný hexagon, který se spojuje se stranou desky pomocí šablony se nazývá kořenový hexagon šablony. Sedm nejpoužívanějších šablon reprezentuje obrázek č. 11.

Červený hexagon šablon znázorňuje kořenový hexagon a šedě označené hexagony představují celkovou rozlohu šablony na desce. Bílý hexagon v šablonách *IIIc* a *IVc* nezapadá do celkové rozlohy šablon. Pokud ani jeden hexagon celkové rozlohy šablony není obsazen protivníkem, nazývá se tato šablona neohrožená a její konektivita je 0.

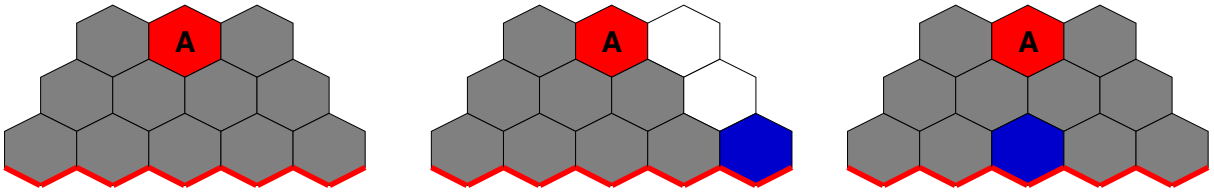


Obrázek 11: Šablony

Princip šablon Šablony jsou označeny pomocí římské číslice a písmena kromě šablony *II*, která není označena písmenem. Číslo šablony určuje vzdálenost kořenového hexagonu od strany

a písmeno definuje verzi šablony. Verze šablony určuje určitý způsob, který umožňuje tento hexagon ke straně připojit. Kámen může být ke straně připojen více šablonami s konektivitou 0 najednou. Pokud protivník napadne celkovou rozlohu alespoň jedné šablony, která bezpečně připojuje kámen ke straně a zároveň je tento kámen ke straně připojován jinou šablonou, jejíž celková rozloha nebyla napadena, zůstává tento kámen bezpečně spojen se stranou.

Levá část obrázku č. 12 zobrazuje červený kámen *A*, který je bezpečně připojen ke straně pomocí šablon *IIIa*, *IIIb* a *IIIc*. Celková rozloha všech těchto šablon je vyznačena šedými hexagony. V prostřední části tohoto obrázku položil modrý hráč kámen na hexagon, který náleží celkové rozloze šablon *IIIa* a *IIIc*, ale nespadá do celkové rozlohy šablony *IIIb* a proto je kámen *A* stále bezpečně připojen ke straně. V pravé části tohoto obrázku položil modrý hráč kámen na hexagon, který náleží celkové rozloze šablon *IIIa* a *IIIb*, ale nespadá do celkové rozlohy šablony *IIIc* a proto zůstává kámen *A* bezpečně připojen ke spodní straně desky.



Obrázek 12: Připojení pomocí více šablon

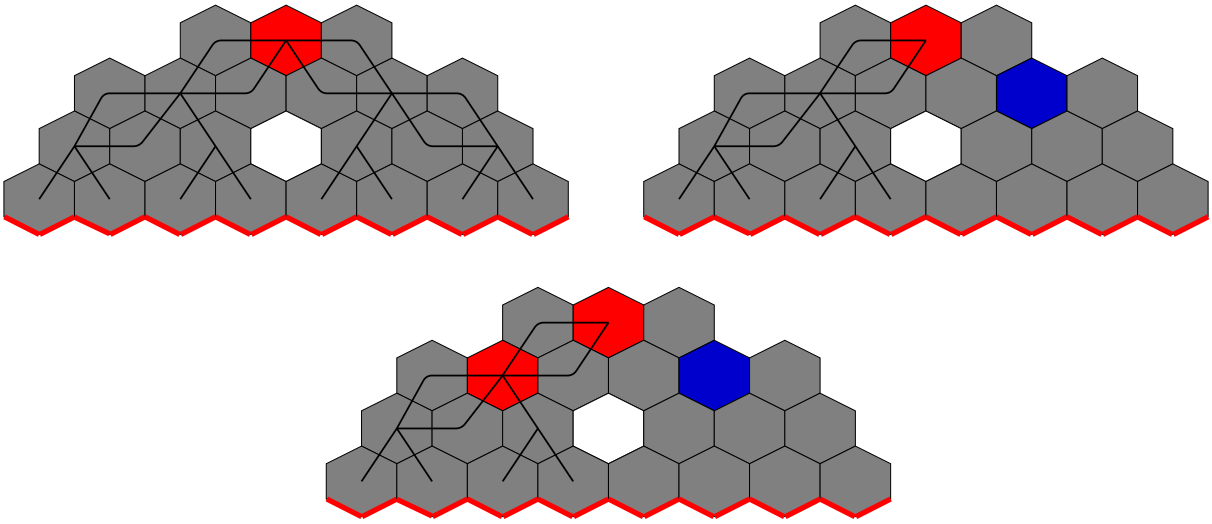
Ať už protivník položí svůj kámen na jakoukoli pozici z celkové rozlohy šablony s konektivitou 0, vždy bude existovat minimálně jeden další způsob, jak lze v dalším tahu zachránit bezpečné připojení ke straně. [4, s. 69] Pro kořenový hexagon totiž existují minimálně dvě od sebe oddělené potenciální cesty ke straně herní desky.

Levá horní část obrázku č. 13 ilustruje průchody největší šablonou - *IVc*. V případě napadení této šablony modrým hráčem (pravá horní část obrázku) dokáže červený hráč obnovit bezpečné spojení kořenového hexagonu ke straně položením kamene na definované místo v šabloně (spodní část obrázku). Nově umístěný kámen bude bezpečně připojen ke kořenovému hexagonu a současně bude kořenovým hexagonem šablony *IIIb*.

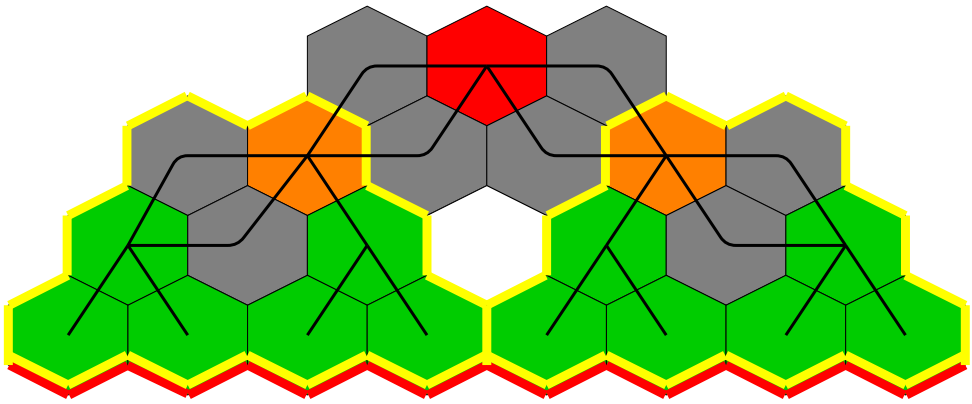
Větší šablony (šablony s větší vzdáleností od strany) pro svou funkčnost využívají menší šablony (šablony s menší vzdáleností od strany). Obrázek č. 14 popisuje rozkreslení šablon, ze kterých se skládá šablona IVc. Šablona *IVc* propojuje svůj kořenový hexagon pomocí dvou mostů k potenciálním novým kořenům šablon *IIIa* a *IIIb*. Kořenové hexagony těchto šablon jsou obarveny oranžově. V každé z těchto podšablon se nachází jeden most a dvě podšablony *II*. Hraniční hexagony celkové rozlohy šablon *IIIa* a *IIIb* jsou pro ilustraci obtaženy žlutou barvou a hexagony šablony *II* jsou zvýrazněny zeleně.

3.1.16 Žebřík

Žebříková struktura vznikne na herní desce, pokud chce hráč bezpečně připojit kameny ke své straně desky, ale protivník jej dokázal zablokovat s dostatečným předstihem a důsledkem bloko-



Obrázek 13: Bezpečné připojení pomocí šablony IVc



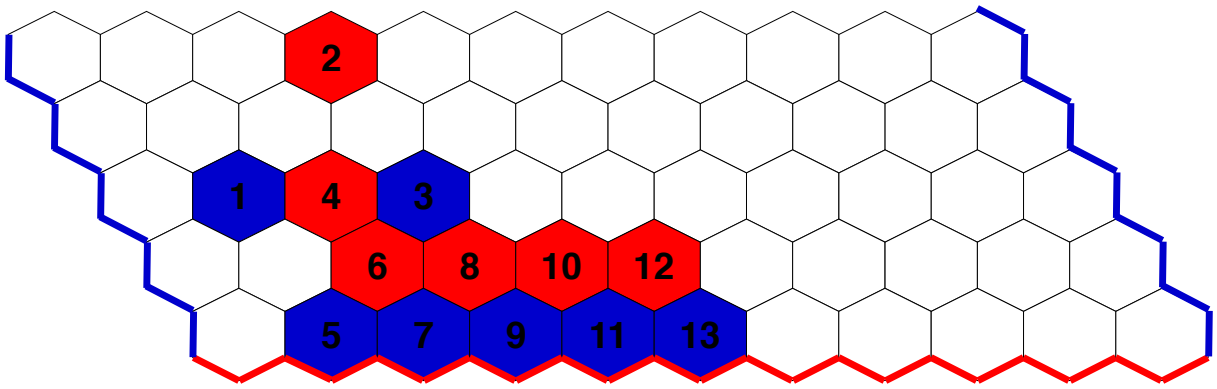
Obrázek 14: Rozdělení IVc na podšablony

vání vznikl na desce řetěz barevných hexagonů v paralelním směru ke straně desky. Tento řetěz se nazývá žebřík.

Tuto situaci červeného hráče popisuje výřez herní desky na obrázku č. 15. Čísla v hexagonech znamenají, v jakém pořadí byly kameny na desku položeny. Pokud se červený hráč, který je na tahu, bude dál marně snažit připojit ke straně desky, modrý hráč tento žebřík zablokuje a zároveň si vytvoří svůj vítězný řetěz. [4, s. 104]

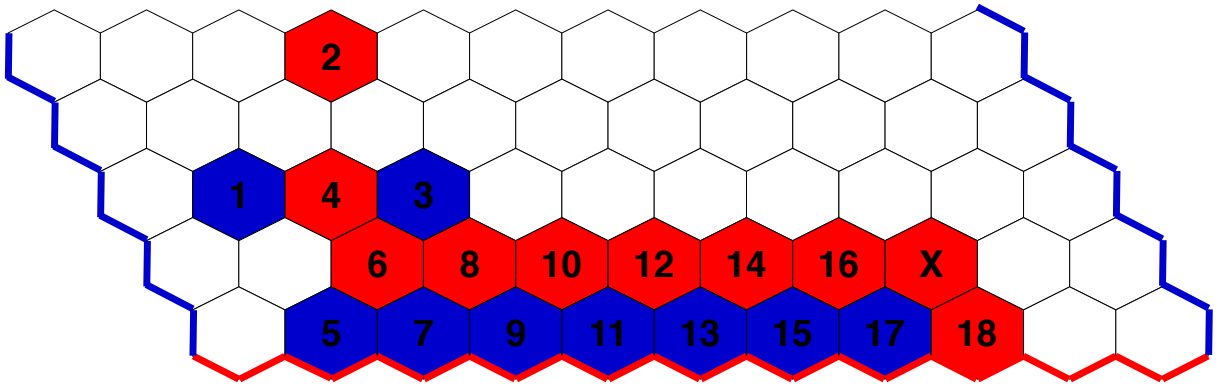
Kameny, které byly na desku umístěné v pořadí jako první a třetí se nazývají tzv. *bottleneck* - česky úzké hrdlo, přes které musí červený hráč projít, aby přiblížil své bezpečně spojené kameny ke své straně. [4, s. 105]

Únik ze žebříku Pokud na herní desce existuje dříve položený vhodně umístěný kámen, který je bezpečně připojený ke straně, je možno z žebříku uniknout přes tzv. únikovou šablonu. Únikové šablony operují s kameny, které se nachází na kraji herní desky a s šablonami. Ukázku



Obrázek 15: Žebřík

úniku z žebříku přes šablonu *II* ukazuje obrázek č. 16. Dříve umístěný červený kámen je označen písmenem X a tento kámen je zároveň kořenovým hexagonem šablony *II*.

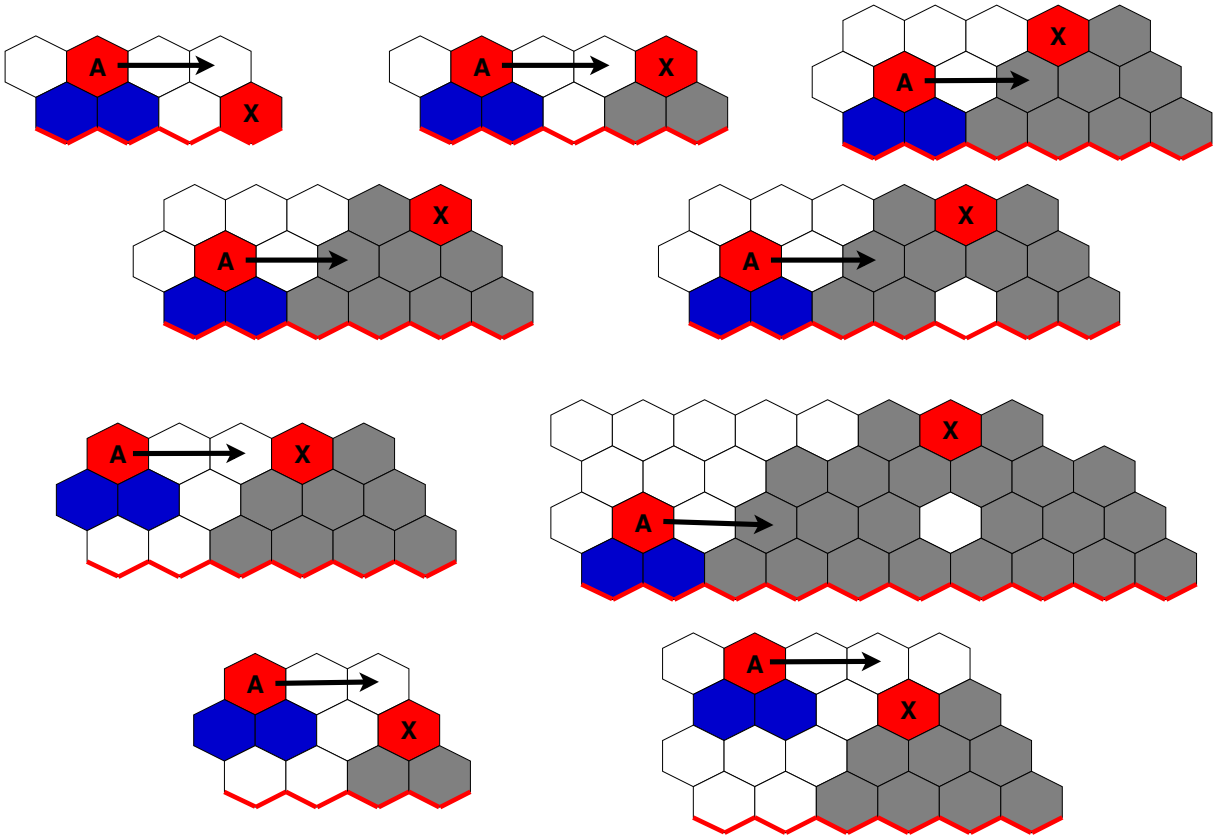


Obrázek 16: Únik z žebříku přes šablonu II

Způsobů, jak lze z žebříku uniknout, existuje více. Některé únikové šablony zobrazuje obrázek č. 17. Dříve umístěný kámen je v šablonách označen písmenem *X*. Šipka v těchto únikových šablonách udává směr, kterým bude žebřík budován z kamene *A*. Pokud dokáže hráč dobudovat žebřík do hexagonu, na který ukazuje šipka, dokázal hráč uniknout z tohoto žebříku. Tyto únikové šablony garantují bezpečný únik z žebříku ke straně za předpokladu, že všechny hexagony celkové rozlohy šablony označené šedou barvou jsou bílé. [15]

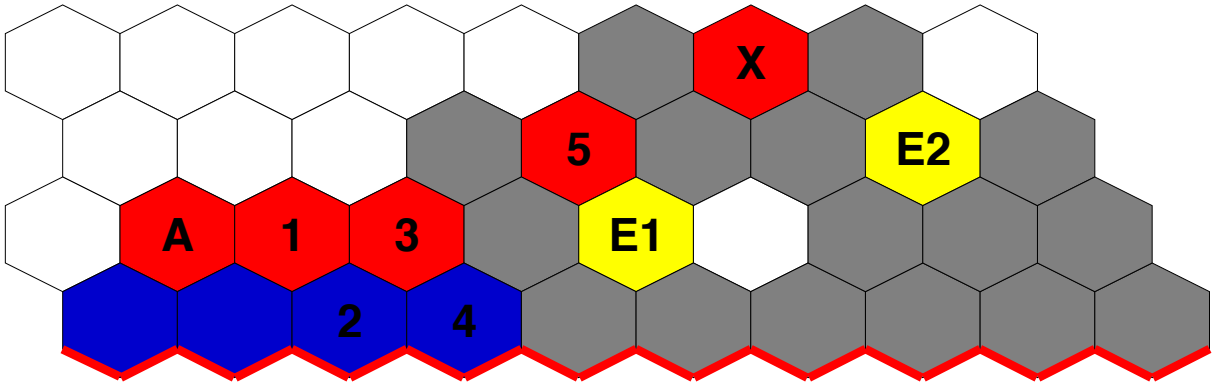
Konkrétní příklad úniku z žebříku od kamene *A* za použití šablony *IVc* zobrazuje obrázek č. 18. Čísla v hexagonech určují pořadí, v jakém byly hexagony obsazeny. Červený hráč dokázal z žebříku uniknout položením kamenu č. *3*. Modrý hráč se snaží ještě marně zablokovat červené kameny od bezpečného spojení ke straně kamenem č. *4*.

Červený hráč v pátém tahu položí kámen na pozici, která bezpečně spojí kámen *A* s kamenem *X*. Žlutě vyznačené hexagony *E1* a *E2* představují dvě potenciální cesty, kterými se dají bezpečně připojit červené kameny ke straně. Položením červeného kamene na pozici *E1* by se tento kámen stal kořenovým hexagonem bezpečně připojené šablony *II* a položením červeného kamene na



Obrázek 17: Únikové šablony

pozici *E2* by se tento kámen stal kořenovým hexagonem bezpečně připojené šablony *IIIa*. Modrý hráč nedokáže obě dvě potenciální cesty zablokovat v jednom tahu.

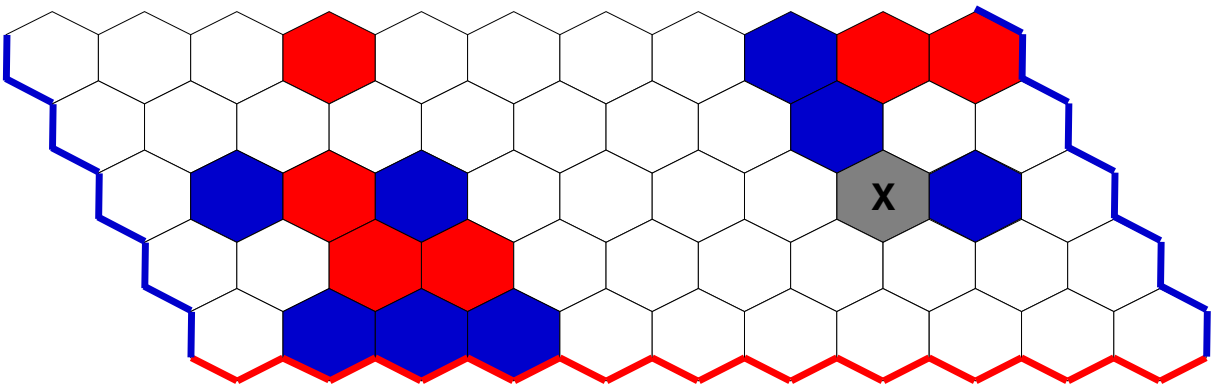


Obrázek 18: Ukázky úniku z žebříku pomocí únikové šablony IVc

Žebříková vidlička Pokud se hráč dostane do situace, že vytvořil žebřík a žádný únik z něj není možný, musí se únik pokusit vykonstruovat. Metoda vytvoření únikové šablony pro žebřík se jmenuje *vidlička*, angl. *fork*. Jedná se o umístění kamene na takovou pozici, aby se nově

umístěný kámen stal kořenovým hexagonem únikové šablony a zároveň by narušoval alespoň jednu bezpečně spojenou protivníkovu strukturu.

Nejjednodušší příklad znázorňuje výřez herní desky na obrázku č. 19. Červený hráč má možnost provést vidličku umístěním svého kamene na pozici označenou písmenem *X*. Tímto tahem si červený hráč vytvoří únikovou šablonu *IIIb* a zároveň ohrozí bezpečně spojený most modrého hráče, který je kritický pro připojení jeho kamenů k pravé straně. Modrý hráč je schopný ve svém tahu reagovat pouze na jednu ze dvou nových hrozeb.



Obrázek 19: Žebříková vidlička

3.1.17 Spojovací cesta

Spojovací cesta je množina bezpečně spojených kamenů, které jsou bezpečně připojeny k oběma stranám hráče. Hráč, který dokáže vytvořit spojovací cestu a v dalších tazích neudělá chybu se stane vítězem. Nejkratší spojovací cesta, která se skládá pouze ze sousedních kamenů je zároveň vítězným řetězem.

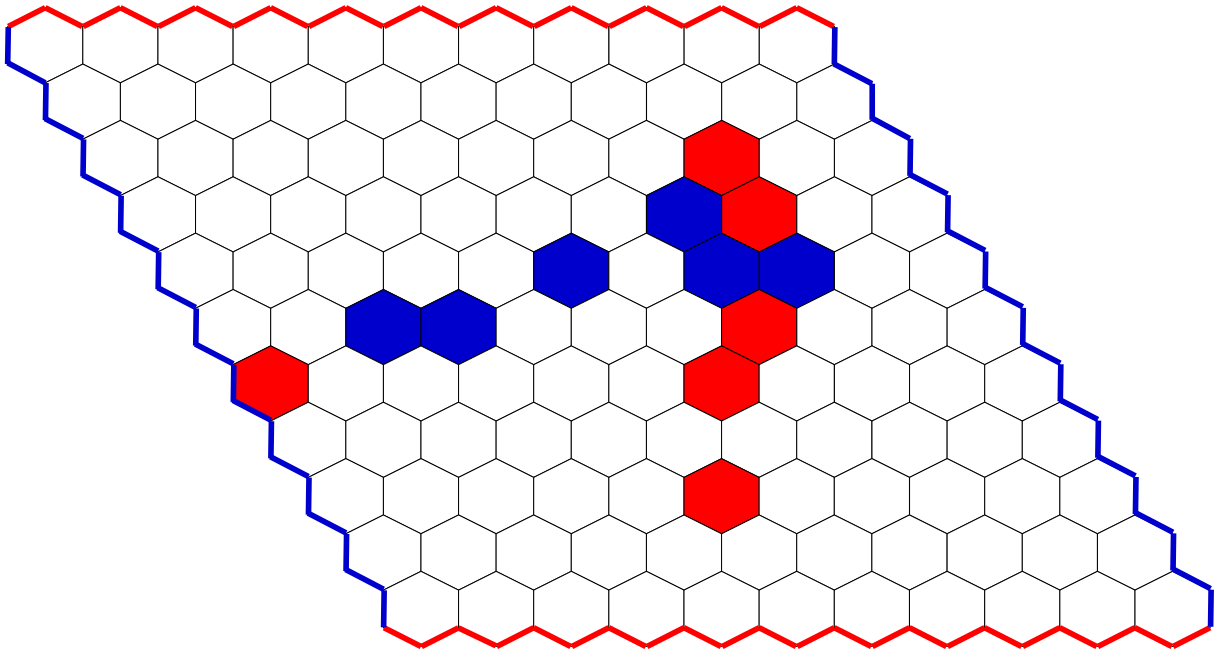
Obrázek č. 20 ilustruje spojovací cestu modrého hráče, která pro bezpečné propojení stran používá mosty, dotýkající se kameny a šablony. [4, s. 64]

3.1.18 Zbytečný kruh

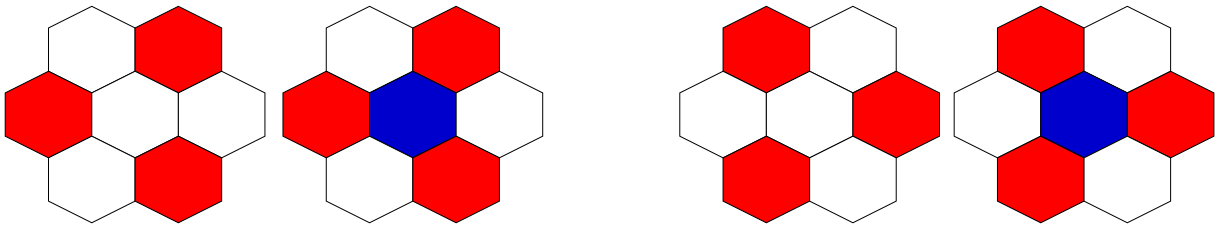
Zbytečný kruh je struktura, která vznikne, pokud na desce existují tři stejně barevné kameny, které zároveň tvoří tři bezpečně spojené mosty a protivník umístí svůj kámen na pozici, která ohrožuje bezpečné spojení všech tří mostů. Obrázek č. 21 znázorňuje dvě možnosti, jak může vzniknout zbytečný kruh červeného hráče. Zbytečný kruh je struktura, které se hráči snaží vyhnout, protože narušuje bezpečné spojení tří mostů.

3.2 Strategie

Tato kapitola popisuje vybrané doporučené strategie, jak postupovat v určitých situacích při hraní hry Hex. Pro pochopení strategií je nutné rozumět pravidlům hry a znát pojmy a struktury. Čísla v kamenech v obrázcích této kapitoly určují pořadí, ve kterém byly tyto kameny položeny.



Obrázek 20: Ukázka spojovací cesty



Obrázek 21: Zbytečné kruhy

3.2.1 Blokování

Blokování znamená položení kamenu na pozici, která znemožní (případně ztíží) protivníkovi vytvořit spojující cestu nebo bezpečně spojit své struktury. [22]

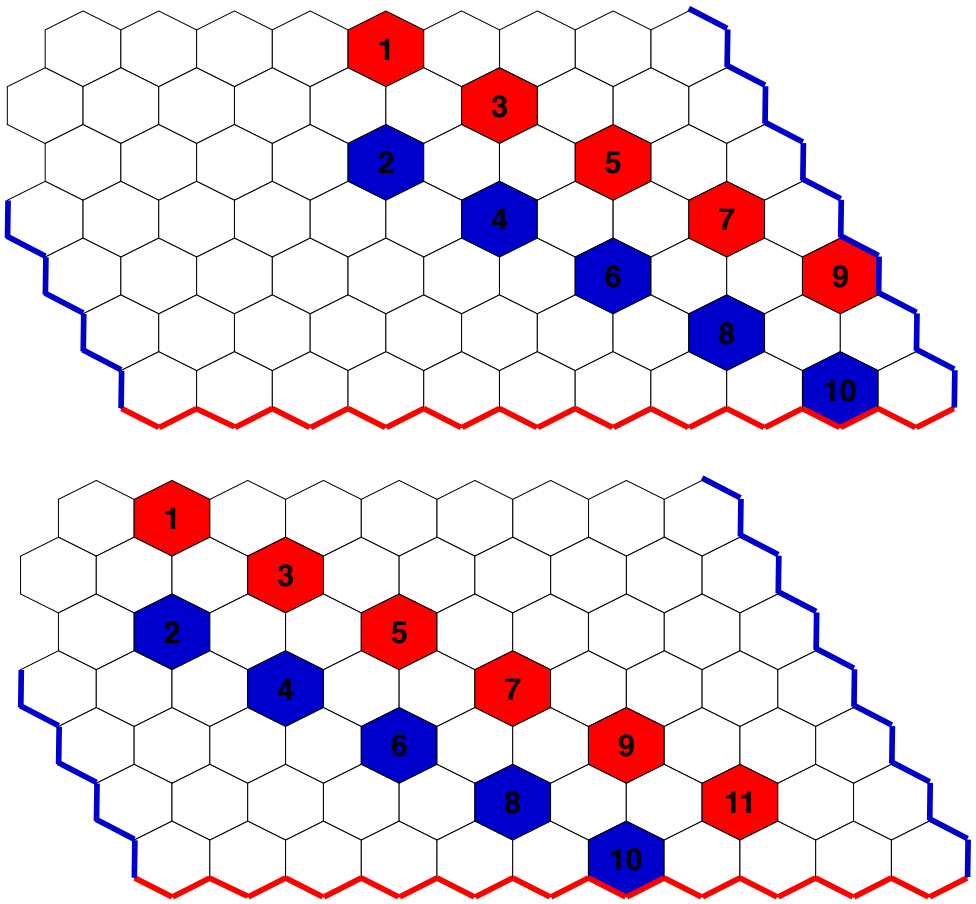
Nejvhodnějšími kandidáty na blokování jsou protivníkovy struktury s konektivitou 1, které by hrály důležitou roli při sestavování jeho spojující cesty.

Za předpokladu, že všechny protivníkovy struktury mají konektivitu 0, nabízí se možnost tzv. blokovat z dálky. Blokování z dálky znamená zabránění vytvoření bezpečného spojení protivníkových kamenů s jeho stranou herní desky.

Blokování z dálky je možné, pokud jsou protivníkovy kameny od strany natolik vzdálené, aby k této straně dosud nebyly bezpečně připojeny. Pro úspěšné blokování z dálky je důležité blokující kámen umístit do dostatečné vzdálenosti.

Obrázek č. 22 ilustruje základní způsob blokování z dálky, který nemusí být efektivní, zejména na velkých herních plánech. Tento způsob tlačí protivníkovy barevné hexagony do ostrého rohu herní desky. Protivník, který se každým tahem přibližuje ke své straně může být tímto způsobem

zablokován, pokud začne blokování s dostatečným předstihem (horní část obrázku). Případ, kdy se tento způsob blokování nevyplatí použít je ukázán ve spodní části tohoto obrázku.



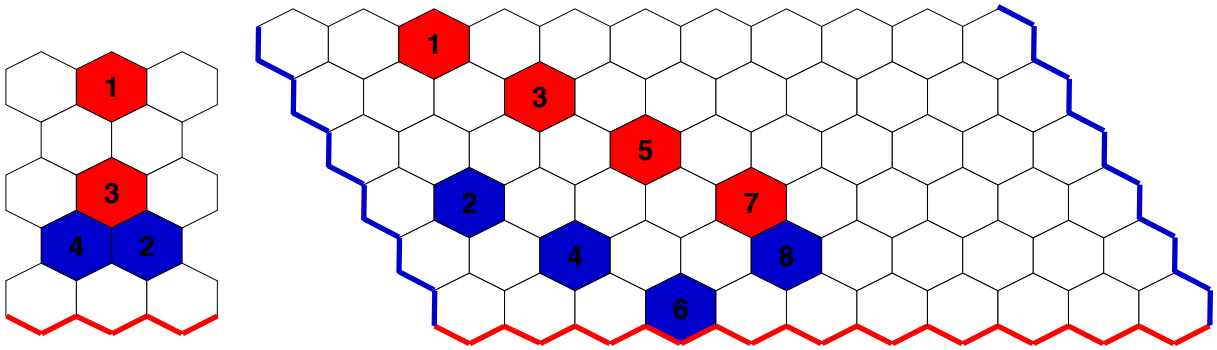
Obrázek 22: Základní blokování z dálky

Efektivní způsob blokování popisuje obrázek č. 23. Levá část obrázku ilustruje případ, který nastane, pokud se červený hráč pokusí umístěním kamenu *3* projít kolem blokujícího protivníkova hexagonu *2*. Přístup kamenů červeného hráče ke straně se zablokuje modrým hexagonem *4*.

Pokud se červený rozhodne k postupnému se přibližování straně pomocí mostů, jako je zobrazeno v pravé části tohoto obrázku, modrý hráč je ho schopen nakonec zablokovat pomocí blokování překrývajících se průchodů, viz kapitola 3.2.2.

3.2.2 Překrývající se průchody

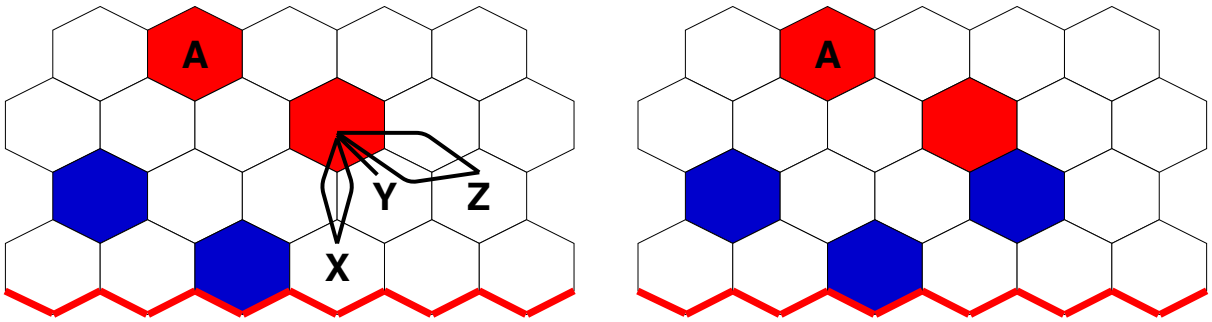
Pokud jsou alespoň dva různé průchody závislé na stejném hexagonu, tak se tyto průchody v tomto hexagonu tzv. *překrývají*. Při blokování se hodí vytipovat si bílý hexagon, ve kterém se překrývají průchody všech plánovaných bezpečných spojení od protivníkova kamene k prvním hexagonům v protivníkových potenciálních cestách ke straně. Obsazením tohoto volného hexagonu, ve kterém se průchody překrývají, se zablokují bezpečné spojení ke všem těmto potenciálním cestám najednou.



Obrázek 23: Efektivní způsob blokování z dálky

Ukázku takové situace zachycuje obrázek č. 24. Předpokládejme, že kámen A je bezpečně připojen k horní straně desky a červený hráč se snaží bezpečně připojit své kameny ke spodní straně. Modrý hráč začal efektivně blokovat z dálky. V levé části obrázku jsou znázorněny průchody, kudy může červený hráč bezpečně připojit své kameny ke spodní straně desky.

Modrý hráč hledá hexagon, ve kterém se překrývají průchody všech bezpečných spojení. Z obrázku lze vyčíst, že se průchody překrývají v hexagonu Y. Položením modrého kamene na tento hexagon se zablokuje potenciální cesta přes Y a bezpečná spojení k hexagonům X a Z (pravá část obrázku).



Obrázek 24: Překrývající se průchody

3.2.3 První tahy

Strategie prvních tahů se liší podle toho, zda se hraje se swapovacím pravidlem.

První tahy bez swapovacího pravidla První hráč ve svém prvním tahu obvykle položí kámen na hexagon, který se nachází ve středu herní desky, nebo jeho blízkém okolí. Na desce o sudé velikosti se nachází dva středové hexagony (ve středu diagonály mezi tupými úhly herní desky). Hexagon nacházející se ve středu herní desky je výhodný, protože vzdálenost k oběma stranám hráče je stejná.

Druhý hráč ve svém prvním tahu má obvykle nevýhodu, protože středový, nebo jemu blízký hexagon na desce byl s největší pravděpodobností obsazen. V tomto případě druhý hráč umístí svůj první kámen na takovou pozici, ze které dokáže začít efektivně blokovat prvního hráče a zároveň tato pozice bude co nejblíže středu herní desky. Pokud středový, nebo jemu blízký hexagon nebyl prvním hráčem obsazen, druhý hráč obsadí jeden z těchto hexagonů.

První tahy se swapovacím pravidlem První tah partie se swapovacím pravidlem musí být natolik slabý, aby nebyl lákavý pro swap modrým hráčem a zároveň by tento tah měl být natolik silný, aby poskytl tzv. výhodu prvního položeného kamenu. Taková výhoda se hodí např. v situaci dříve položeného kamenu při úniku z žebříku viz kapitola 3.1.16. První kámen se obvykle pokládá na hexagon, který je sousedící se stranou herní desky, nebo který se nachazí v její blízkosti.

Pokud první položený kámen neleží blízko středu desky nebo o něj modrý hráč nemá zájem, obvyklý první tah modrého hráče bývá položení svého kamenu na středový hexagon bez swapování. Obecně je doporučeno swapovat všechny pozice kromě těch, které jsou sousedící se stranou herní desky, nebo které se nachází v její blízkosti. [4, s. 153]

3.2.4 Nutící tahy

Pokud položí hráč na desku svůj kámen a donutí tím protivníka k nutné odpovědi, provedl hráč tzv. nutící tah. Nutná odpověď může být obrana své struktury nebo blokování protivníka od vytvoření spojující cesty. Typickým nutícím tahem bývá žebříková vidlička popsaná v kapitole 3.1.16. Nutící tahy se používají k získání momenta.

3.2.5 Rozšiřování

Rozšiřování znamená provedení kroku od kamene skupiny směrem ke straně, která není ke kamenům skupiny bezpečně připojena. Pro nejefektivnější rozšiřování se hodí budovat bezpečně spojené mosty, pokud je to možné. Mosty totiž pokryjí dvojnásobnou vzdálenost na desce oproti sousedním kamenům. Ideální tah je ten, který dokáže vhodně rozšířit skupinu a zároveň bude blokovat protivníka. Při rozšiřování má hráč momentum.

4 Popis vlastní implementace

Tato kapitola se zabývá popisem vlastní implementace hry Hex. Celý vlastní program má název *HexGame*. Uživatelská příručka programu se nachází v příloze A.

4.1 Použité knihovny a nástroje

HexGame je napsaný v jazyce C# 5.0 pro verzi .NET Framework 4.6.1. Program byl vyvíjen v Microsoft Visual Studiu 2017 Community edition na OS Windows 10 64-bit Home edition. Při vývoji nebyly použity žádné knihovny třetích stran. HexGame používá třídy těchto jmenných prostorů z .NET Frameworku:

- System
- System.Windows.Forms
- System.Drawing
- System.Collections.Generic
- System.Collections.ObjectModel

Podrobnější dokumentace vlastního kódu se nachází v příloze A této práce. Třídní diagram a Code Map byly vytvořeny pomocí Visual Studia. Závislostní diagram byl vytvořen za pomocí softwaru StarUML. Dokumentace kódu byla vygenerována pomocí XML komentářů v kódu a rozšíření Sandcastle Help File Builder pro Visual Studio.

4.2 Struktura programu

V následující části je popsána architektura, komunikace a vybrané algoritmy vlastní implementace.

4.2.1 Architektura

Architektura programu je lineární dvouvrstvá. Vrstvy architektury zobrazuje obrázek 25. Šipky v obrázku představují směr komunikace.

První vrstva je prezentační. Jedná se o GUI - desktopovou Windows Form aplikaci. Tato vrstva slouží jako grafické rozhraní ke komunikaci uživatele a doménové vrstvy programu. Prezentační vrstva se stará o vykreslování herní desky, zpracování vstupu uživatele a zasílání požadavků do doménové vrstvy, viz kapitola 4.2.3.

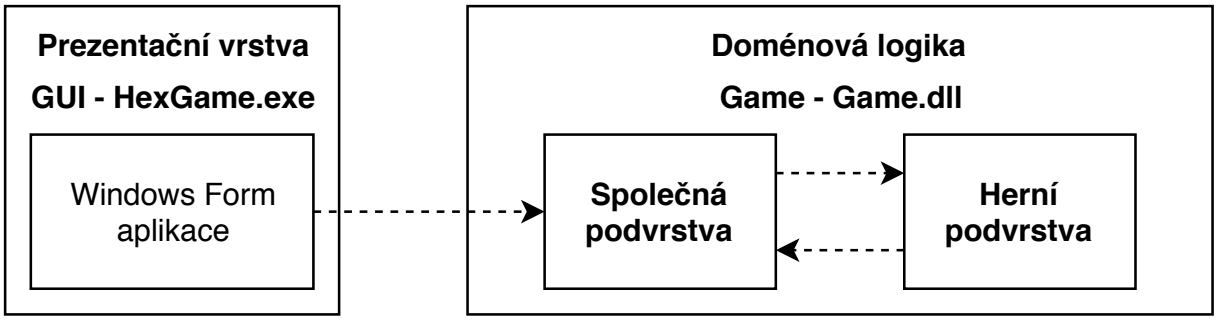
Druhá vrstva - doménová logika - je zabalena do jedné knihovny tříd. Tato vrstva je rozdělena na dvě podvrstvy. První, tzv. společná podvrstva, obsahuje data, ke kterým přistupují všechny části programu. Na tomto místě se soustřeďují veškeré informace, definice a data, které musí

být veřejně dostupné. Jedná se např. o definici Hexagonu, globální proměnné, herní desku a její velikost, algoritmy pro hledání vítězného řetězu hexagonů a jiné.

Druhou podvrstvou doménové logiky je herní podvrstva. Zde se nacházejí definice herních struktur a implementace umělé inteligence. Tato podvrstva přistupuje do společné podvrstvy, odkud získává data o herní desce, na které se algoritmy spouští.

Třetí vrstva není potřeba, data není nutné ukládat, ani je načítat. Z tohoto důvodu stačí pouze dvě vrstvy.

Prezentační vrstva je oddělená od ostatní logiky programu. Naimplementované algoritmy i společné data jsou nezávislé na této vrstvě. V případě potřeby jiného klienta by stačilo naprogramovat jinou aplikaci prezentační vrstvy, která by stejným způsobem přistupovala ke společným datům.



Obrázek 25: Architektura programu

4.2.2 Herní deska

Základem správné implementace je mít dobrý základ - herní desku, ke které je možno přistupovat více způsoby. Na herní desce se pak odehrávají veškeré analýzy a spouští se zde naimplementované algoritmy. Existuje několik způsobů, jak lze herní plán reprezentovat.

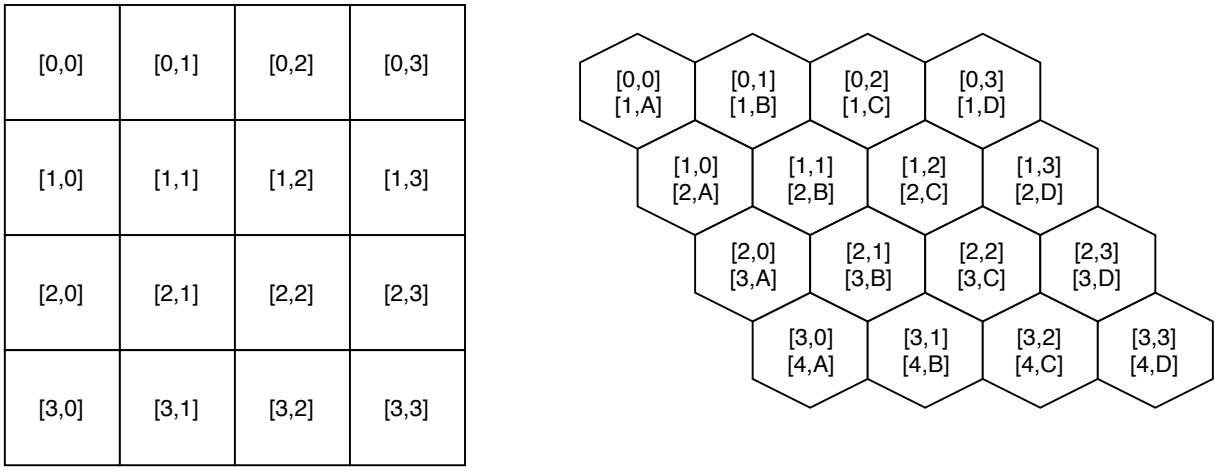
První způsob reprezentace herní desky je dvourozměrné pole. Jelikož se inicializuje herní deska o rozměru $n * n$, kde n je velikost desky, nabízí se možnost udržovat informace o herním plánu v paměti jako dvourozměrné pole hexagonů. Barva prvku pole určuje barvu hexagonu na herním plánu. Každý hexagon je jasně definován adresou řádku a sloupce v poli. Tento způsob reprezentace herní desky má výhodu, že se celá deska dá vnímat jako jedno dvourozměrné pole.

Druhý způsob reprezentace herní desky je graf - množina vrcholů a hran. Vrcholy grafu jsou hexagony, které jsou spojeny hranou, pokud spolu sousedí na herní desce. Veškeré hrany jsou obousměrné a mají stejnou váhu. V tomto případě je možno vytvořit si množinu hexagonů, ve které má každý hexagon definované sousedy (viz kapitola 4.2.3). Hexagon má v sobě uložené ukazatele na své sousední hexagony. Není problém snadno zjistit, jestli se daný hexagon nachází na okraji desky - ukazatele na příslušné sousedy by v tom případě byly nastaveny na *null*.

Vlastní implementace používá kombinaci obou reprezentací herní desky. Při inicializaci desky se vytvoří dvourozměrné pole, které se naplní novými hexagony, do kterých se v konstruktoru

vloží parametry, které určují umístění v poli. Každý hexagon má v sobě uloženou adresu v poli pomocí dvou proměnných *row* a *col*, které udávají hodnotu indexu řádku a sloupce v poli. Dále každý hexagon obsahuje šest ukazatelů na sousední hexagony a barvu tohoto hexagonu. Tento kombinovaný přístup umožní uchovat výhody obou reprezentací herní desky popsaných výše.

Vizualizaci uložení herní desky o velikosti 4 v paměti představuje obrázek č. 26. Na levé straně je zobrazena inicializace pole bez sousedních ukazatelů. Na pravé straně je možno vidět uložení hexagonů v poli po nastavení sousedních ukazatelů. I když je daný hexagon v paměti v poli uložen na adrese [0,0], pro vizualizaci tahů a desky se používá adresa [1,A]. Veškeré algoritmy pracují pouze s hodnotou [0,0]. V levé části obrázku je také jasně vidět, že kdyby se použil pouze přístup z pole, tak na první pohled není jasné, které hexagony jsou spolu sousedící. Při kliknutí na herní plán v GUI se přepočítávají souřadnice myši na korespondující pozici hexagonu v poli.



Obrázek 26: Vizualizace herní desky v paměti

4.2.3 Třídy a dědičnost

V této části jsou stručně představeny vybrané třídy doménové logiky programu. Společná podvrstva obsahuje pouze dvě třídy. Jedná se o třídy *Globals* a *Hexagon*.

Hexagon Třída Hexagon definuje jeden šestiúhelník na herní desce. V ukázce kódu 1 je možno nahlédnout na nejdůležitější část definice třídy Hexagon. Properties *TopR, LowerR, L, LowerL, R* a *TopL* slouží jako ukazatele na sousední Hexagony. Pokud je ukazatel *null*, znamená to, že Hexagon nemá tohoto souseda, tudíž se nachází na okraji herní desky. Přesnou pozici v poli určují properties *ROW* a *COL*. Diskuse nad způsoby uložení herní desky se nachází v kapitole 4.2.2. Barva Hexagonu se určuje pomocí property *COLOR*. Property *Preceeding* se používá pro udržení ukazatele na předchozí hexagon při běhu některých algoritmů průchodů grafem herní desky.

V této třídě je také definován výčtový typ *ColorDefinition*, který definuje možné barvy hexagonu a to *Red, Blue, White, Green* a *Yellow*. Červená a modrá barva jsou barvy dvou

```
public class Hexagon
{
   public Hexagon TopR { get; set; }
   public Hexagon LowerR { get; set; }
   public Hexagon L { get; set; }
   public Hexagon LowerL { get; set; }
   public Hexagon R { get; set; }
   public Hexagon TopL { get; set; }

   int _row;
   int _col;
   public int ROW { get { return _row; } }
   public int COL { get { return _col; } }

   public enum ColorDefinition { Red, Blue, White, Green, Yellow }
   public ColorDefinition Color { get; set; }

   public Hexagon Preceeding { get; set; }
}
```

Výpis 1: Definice třídy Hexagon

hráčů, bílá značí neobsazený hexagon. Žlutá barva se používá pro vykreslení tipu AI na desce na prezentační vrstvě a zelenou barvou se označuje nejkratší výherní řetěz kamenů jednoho z hráčů.

Globals Třída Globals je statická třída, ve které se nachází komunikační rozhraní pro komunikaci mezi prezentační a doménovou vrstvou a data, které musí být veřejně přístupné pro celou doménovou vrstvu.

Výpis kódu č. 2 umožňuje náhled na nejdůležitější části kódu této třídy. Celá herní deska je zde uložena jako property *Board*. Property *Size* určuje velikost herní desky a property *BlueTurn* určuje, zda je na řadě modrý (true), nebo červený hráč (false). Tyto properties jsou používány algoritmy herní podvrstvy i aplikací prezentační vrstvy pro získání informací o herní desce při vykreslování na GUI.

Dále se v této třídě nacházejí statické metody, které slouží pro komunikaci aplikace prezentační vrstvy s doménovou vrstvou. Tyto metody komunikačního rozhraní se používají následovně:

- *InitBoard* slouží k inicializaci nové herní desky o určité velikosti.
- *PlaceHexagon* se volá pro umístění kamene na herní desku, když uživatel klikne na volný hexagon v uživatelském rozhraní.
- *EndOfTurn* se volá po dokončení každého tahu.

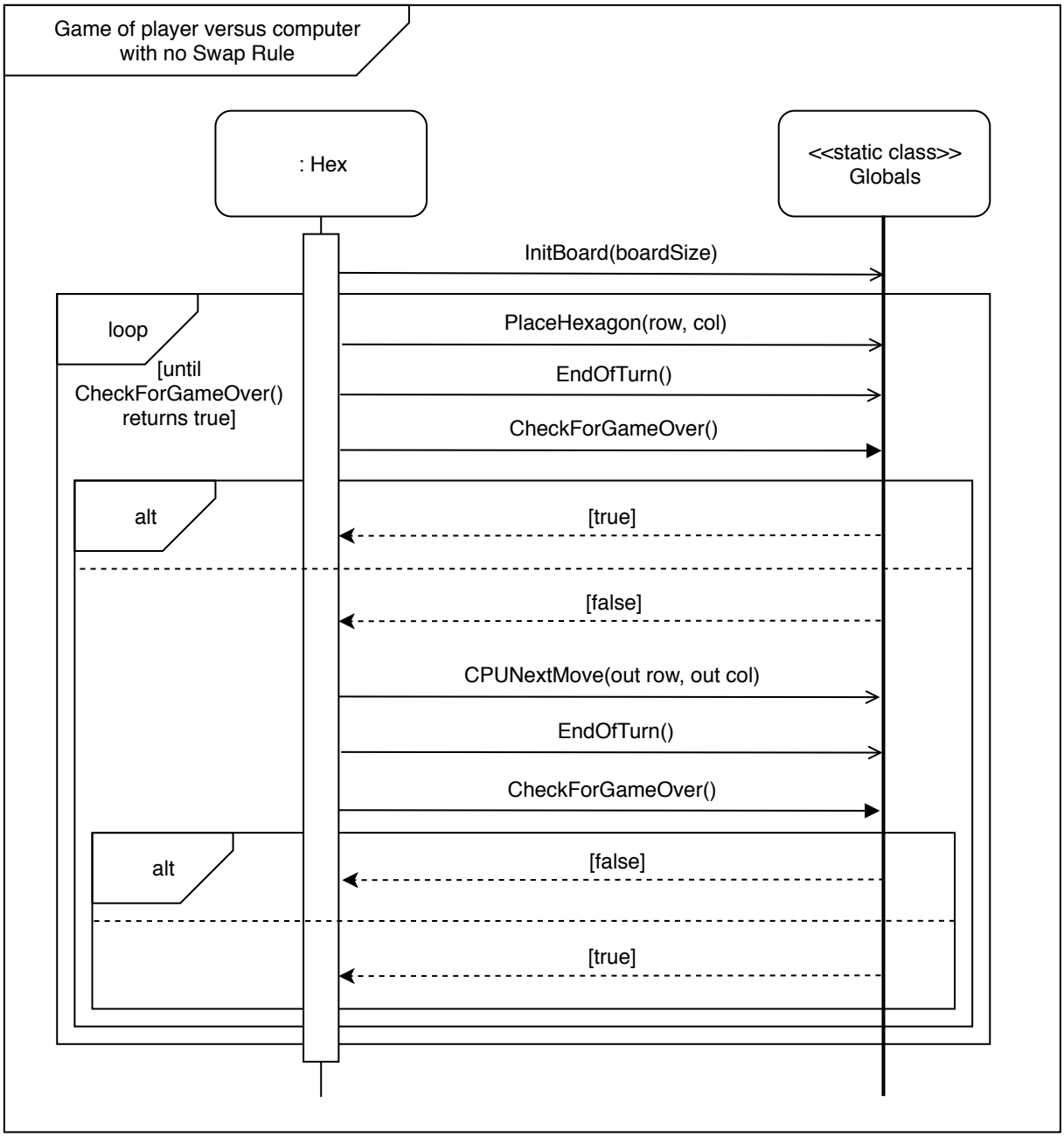
```
public static class Globals
{
   public static int Size { get; internal set; }
   public static bool BlueTurn { get; internal set; }
   public static Hexagon[,] Board { get; internal set; }

   public static void InitBoard(int boardSize);
   public static void PlaceHexagon(int row, int col);
   public static void EndOfTurn();
   public static bool CheckForGameOver();
   public static void CPUNextMove(out int row, out int col);
   public static void CPUNextMoveCanSwap(out int row, out int col);
   public static void CPUHint(out int row, out int col);
   public static void CPUHintCanSwap(out int row, out int col);
}
```

Výpis 2: Ukázka z třídy Globals

- *CheckForGameOver* slouží k prohledání herní desky, zda existuje řetěz, který propojuje dvě strany. Pokud byl řetěz nalezen, tato metoda přebarví kameny tohoto řetězu zelenou barvou.
- *CPUHint* hledá pozici bílého hexagonu, která slouží jako tip od AI, kam umístit kámen v tomto tahu. Tato metoda pouze vrací souřadnice bílého hexagonu a nijak nezasahuje do herní desky. Tato metoda se volá, pokud se nehraje se swapovacím pravidlem.
- *CPUHintCanSwap* hledá pozici bílého hexagonu, která slouží jako tip od AI, kam umístit kámen v tomto tahu. Tato metoda pouze vrací souřadnice bílého hexagonu a nijak nezasahuje do herní desky. Tato metoda se volá, pokud se hraje se swapovacím pravidlem.
- *CPUNextMove* umístí kámen na herní desku na pozici, která byla nalezena pomocí metody *CPUHint*. Tato metoda vrací souřadnice hexagonu na herní desce, na který byl položen kámen. Tato metoda se volá, pokud se nehraje se swapovacím pravidlem.
- *CPUNextMoveCanSwap* umístí kámen na herní desku na pozici, která byla nalezena pomocí metody *CPUHintCanSwap*. Tato metoda vrací souřadnice hexagonu na herní desce, na který byl položen kámen. Tato metoda se volá, pokud se hraje se swapovacím pravidlem.

Komunikace mezi prezentační a doménovou vrstvou Sekvenční diagram na obrázku č.27 zobrazuje průběh komunikace mezi prezentační a doménovou vrstvou při partii hráče proti počítači bez swapovacího pravidla.



Obrázek 27: Komunikace prezentační a doménové vrstvy

Herní podvrstva Herní podvrstva obsahuje tříd mnohem více. Kód je rozdělen do dvou složek *Structures* a *Strategies*. Složka *Structures* obsahuje definice tříd herních struktur popsaných v kapitole 3.1 - most, dvojitý most, skupina, šablony, žebřík, volné spojení a zbytečný kruh. Složka *Strategies* obsahuje statické třídy, ve kterých jsou uloženy algoritmy pro strategie vysvětlené v kapitole 3.2, evaluaci desky a funkce pro práci se strukturami. Vybrané metody těchto statických tříd jsou volány ze třídy *Globals*. Důležitá je zde statická třída *Shared*, která obsahuje obecné sdílené definice a algoritmy, které jsou používány více třídami kódu herní vrstvy.

```
public static class Shared
{
   public enum BlueTemplateDirection { TO_LEFT, TO_RIGHT };
   public enum RedTemplateDirection { TO_TOP, TO_BOTTOM };
}
```

Výpis 3: Výčtové typy k určení směru ke straně desky

Dědičnost a šablony Za zmínku stojí popsat, jak jsou implementovány šablony. Narozdíl od ostatních struktur, které si nejsou ničím podobné, všechny šablony se používají stejně nezávisle na tom, o kterou šablonu jde. Tohohle poznatku lze v implementaci využít pomocí dědičnosti a rozhraní. Složka *Structures* obsahuje definici rozhraní *ITemplate*. Tohle rozhraní definuje hlavičky funkcí, které musí implementovat každá šablona. Z tohohle rozhraní dědí další dvě rozhraní a to *IRedTemplate* a *IBlueTemplate*. Třídy, které realizují tyto dvě rozhraní už obsahují implementaci šablon samotných. Tyto dvě dodatečná rozhraní byla zavedena, aby bylo jednoznačně možné určit, ke které barvě strany herní desky šablona směřuje. Z tohoto důvodu jsou zavedeny další dva výčtové typy, které se dají také použít ke konkrétnímu určení strany herní desky. Výčty směrů ke straně desky ukazuje výpis č. 3. Ukázku dědičnosti a rozhraní popisuje třídní diagram na obrázku č. 28. I když jsou pro ilustraci uvedeny pouze dvě konkrétní šablony každé barvy, vlastní implementace pracuje se všemi sedmi šablonami popsanými v kapitole 3.1.15.

4.3 Použité algoritmy ve vlastním programu

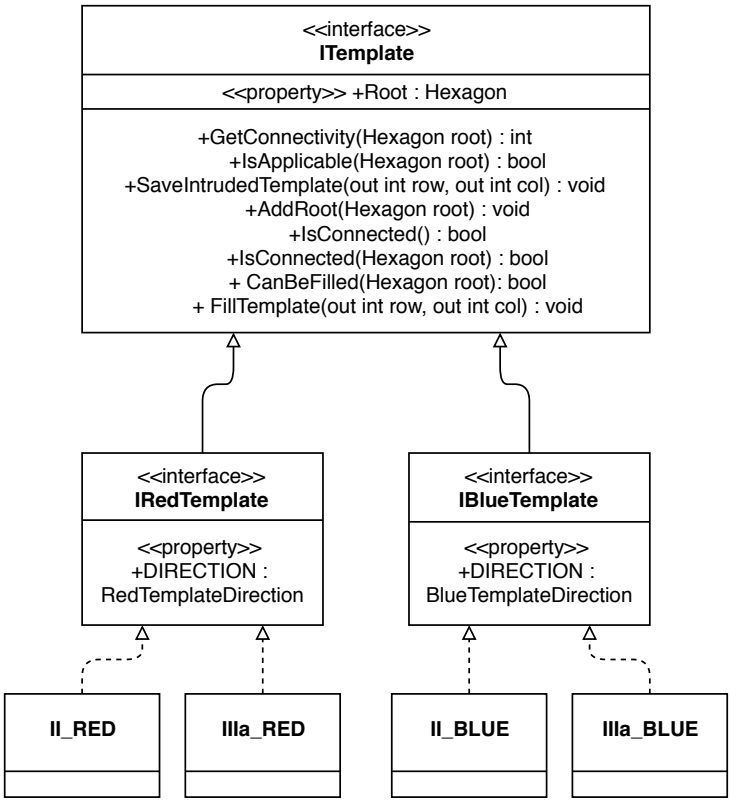
V této části budou popsány nejdůležitější a zajímavé algoritmy se kterými pracuje vlastní implementace.

4.3.1 Průchod grafem do šířky

Průchod grafem do šířky (angl. breadth-first search) se používá k nalezení nejkratší cesty od jednoho kamene ke druhému pouze přes sousední stejně barevné hexagony. BFS se v implementaci využívá pro nalezení nejkratšího výherního barevného řetězu hexagonů. BFS je možno použít pro hledání nejkratších cest, protože jsou všechny hrany grafu stejně ohodnoceny. [18]

4.3.2 Průchod grafem do hloubky

Průchod grafem do hloubky (angl. depth-first search) se v implementaci využívá pro nalezení všech barevných hexagonů tvořící jeden řetěz. Řetěz, který je nalezen pomocí DFS má jiné pořadí nalezených prvků než řetěz nalezený pomocí BFS. Použití DFS je jednodušší, protože pracuje pouze s množinou navštívených hexagonů, zatímco BFS musí pracovat i s frontou hexagonů k navštívení pro udržení pořadí průchodu. [17]



Obrázek 28: Dědičnost šablon

4.3.3 Bezpečný průchod grafem do šířky

Tento průchod grafem (angl. Spanning-BFS) se od BFS liší způsobem nalezení nejkratší cesty. BFS prohledává graf pouze přes sousední kameny, zatímco SBFS prohledává graf i za pomocí neohrožených mostů. BFS nenajde žádnou existující cestu v situaci, kdy jsou dva kameny bezpečně propojeny, ale nejsou součástí jednoho řetězu. V těchto případech se musí použít SBFS.

SBFS se v implementaci používá pro hledání nejkratších existujících cest, nalezení kamenů ve skupině a určení, zda jsou dva kameny nebo kámen a strana spolu bezpečně propojeny. V případě určení, zda jsou kámen a strana bezpečně propojeny, SBFS používá i šablony.

4.3.4 Smíšený průchod grafem

Pro potřeby vlastní implementace hry byl vytvořen nový, tzv. smíšený způsob průchodu grafem, který vznikl modifikací SBFS. Tento způsob byl nazván Semi-spanning-breadth-first search. Tento průchod grafem se ve vlastní implementaci používá pro určení nejkratší potenciální cesty mezi bezpečně spojenými kameny skupiny a stranou.

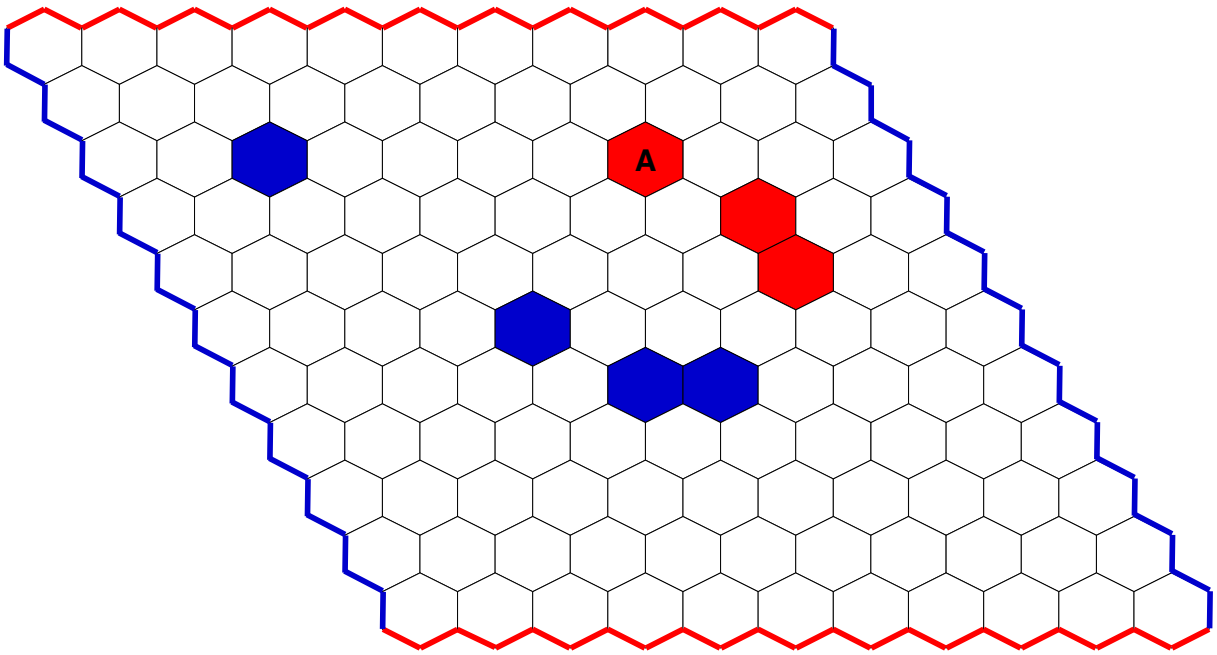
Požadované výsledky tohoto průchodu grafem jsou množina hexagonů určující nejkratší potenciální cestu a její konektivita. Pokud je konektivita kamenu a strany rovna nule, nejkratší potenciální cesta nebude existovat, protože jsou bezpečně spojeny.

BFS a DFS jsou metody průchodů grafem, které prochází přes kameny v řetězu. SBFS je metoda průchodu grafem, která prochází přes kameny ve stejné skupině. BFS, DFS i SBFS jsou tedy nepoužitelné v případech, kdy ještě není vybudována existující cesta mezi kamenem a stranou. SemiSBFS prochází primárně přes bezpečně spojené kameny a pokud nebyla nalezena existující cesta ke straně, dokáže tento algoritmus nalézt nejkratší potenciální cestu z těchto bezpečně propojených kamenů.

SemiSBFS se liší od SBFS několika způsoby. Narozdíl od SBFS, který využívá frontu pro uložení hexagonů k navštívení, SemiSBFS používá spojový seznam, který funguje jako prioritní fronta. Na začátek tohoto seznamu se přidávají bezpečně spojené barevné hexagony a na konec seznamu se připojují bílé hexagony, které jsou dosažitelné v jednom kroku z aktuálně navštíveného hexagonu. Tím se dosáhne požadavku, aby SemiSBFS průchod primárně procházel barevné hexagony a bílé až sekundárně. SemiSBFS prochází primárně přes barevné hexagony, aby se při hledání potenciální cesty využily již položené kameny a tudíž by počet plánovaných kroků byl co nejmenší.

Pokud je první prvek ve spojovém seznamu barevný hexagon, přistoupením na něj se využívá průchod přes již položený kámen. Pokud je první prvek ve spojovém seznamu bílý hexagon, přistoupením na něj se plánuje krok. Algoritmus pokračuje, dokud nenalezne nejkratší potenciální cestu, nebo dokud nebude spojový seznam prázdný. V takovém případě nebude potenciální cesta existovat.

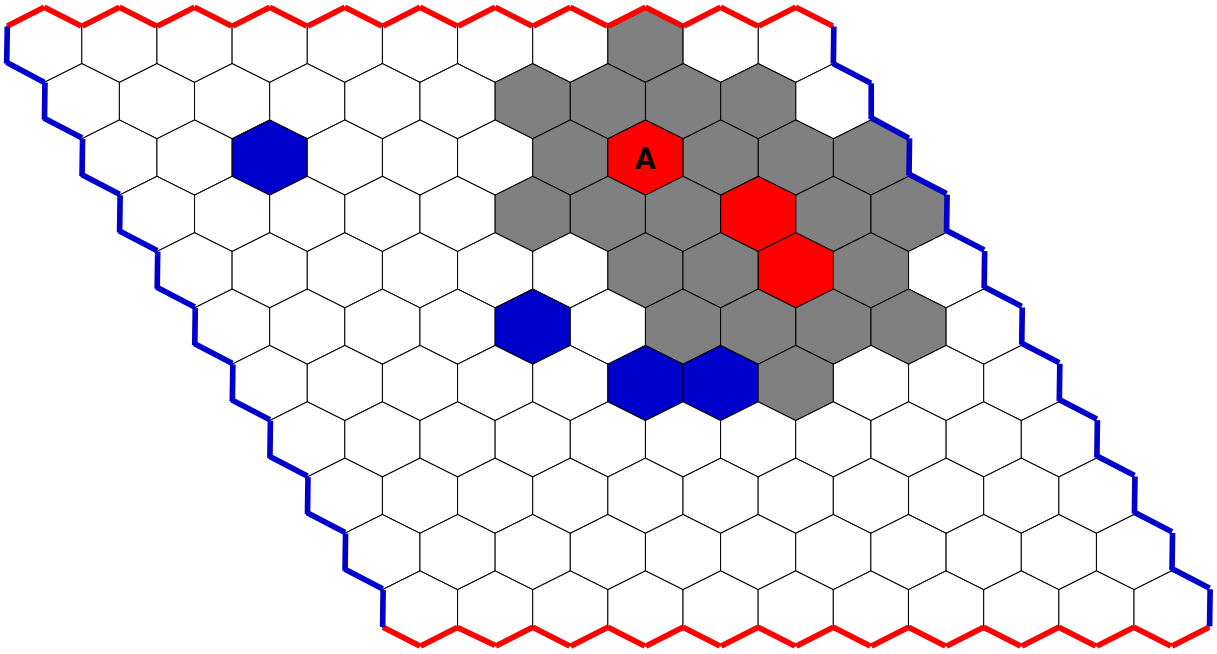
Vhodný případ pro vysvětlení SemiSBFS ilustruje obrázek č. 29. SBFS průchod nenašel bezpečné spojení mezi kamenem A a spodní stranou desky, tudíž je potřeba spočítat jejich konektivitu pomocí SemiSBFS.



Obrázek 29: Vhodný případ pro použití smíšeného průchodu grafem

Na obrázku č. 30 jsou šedou barvou označeny všechny hexagony, na které je možno přistoupit pomocí jednoho kroku z kamenů skupiny, ve které se nachází kámen *A*. Tyto hexagony jsou zároveň hexagony z množiny bílých hexagonů této skupiny. Pokud obsazením jednoho z těchto šedých hexagonů vznikne bezpečné spojení mezi *A* a spodní stranou desky, je tento hexagon zároveň nejkratší potenciální cestou.

SemiSBFS postupně na tyto hexagony vstoupí a ověří, zda případným položením kamenu na tuto pozici vznikne bezpečné spojení mezi kamenem *A* a spodní stranou desky. Pokud ano, algoritmus končí. Pořadí, ve kterém algoritmus přistupuje k šedým hexagonům je určeno podle pořadí vkládání bílých hexagonů na konec spojového seznamu.

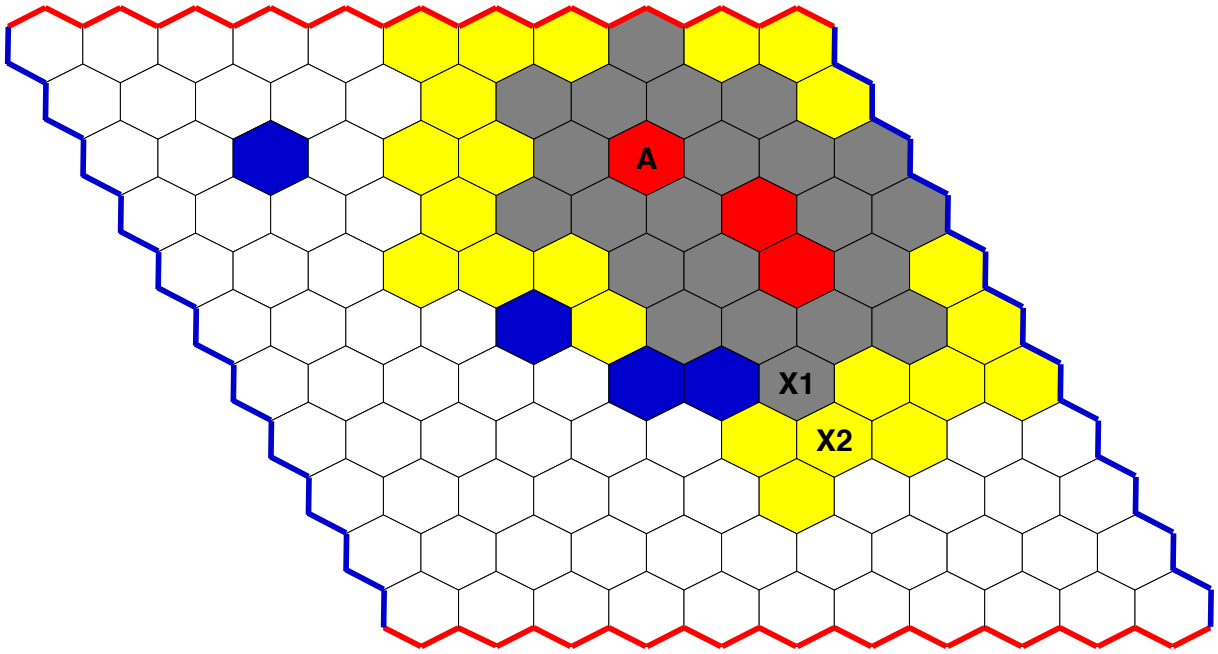


Obrázek 30: Plánování prvních kroků ze skupiny pomocí SemiSBFS

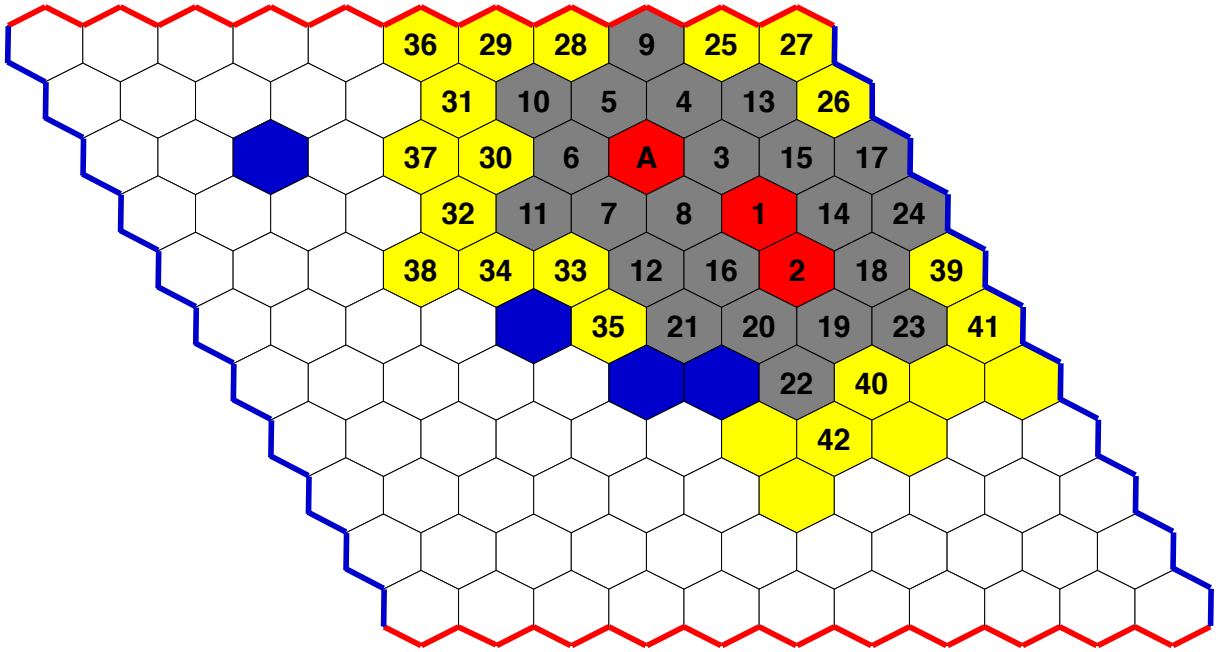
Potenciální cesta od kamene *A* ke straně nebyla pomocí plánování jednoho kroku nalezena. SemiSBFS musí pokračovat plánováním dalších kroků. Na obrázku č. 31 jsou žlutou barvou označeny hexagony, na které se dá přistupit pomocí dvou kroků z barevných hexagonů skupiny, ve které se nachází kámen *A*. Algoritmus opět postupně prochází tyto žluté hexagony.

Algoritmus nalezl nejkratší potenciální cestu, která se skládá z hexagonů *X1* a *X2*. Kámen položený na pozici *X1* by byl bezpečně připojený ke kamenům skupiny pomocí neohroženého mostu a kámen položený na pozici *X2* by se dotýkal kamene na pozici *X1* a zároveň by byl bezpečně připojen ke straně pomocí šablon *IVa, IVb* a *IVc*. Konektivita kamene *A* a spodní strany je tedy 2.

Pořadí, v jakém SemiSBFS přistoupí na hexagony herní desky zobrazuje obrázek č. 32 za předpokladu, že se herní deska prochází od kamene *A*. Algoritmus končí, když vstoupí na 42. hexagon.



Obrázek 31: Plánování druhých kroků ze skupiny pomocí SemiSBFS



Obrázek 32: Pořadí přístupu k hexagonům v SemiSBFS

Popis algoritmu Algoritmus je rekurzivní. Vstupní parametry SemiSBFS algoritmu jsou:

- visited - Kolekce hexagonů, které již byly navštíveny
- toVisit - Spojový seznam hexagonů, které se mají navštívit

- direction - Strana, ke které se hledá potenciální cesta. Barva strany udává barvu hexagonů, přes které se má primárně procházet.

Algoritmus má jeden výstupní parametr.

- path - Spojový seznam, který po skončení algoritmu bude obsahovat bílé hexagony potenciální cesty

Pro úspěšné spuštění algoritmu je nezbytné, aby se ve spojovém seznamu *toVisit* nacházel právě jeden hexagon a to hexagon s nastavenou property *Preceeding* na *null*. SemiSBFS se tzv. *spouští od tohoto hexagonu*. Jednotlivé kroky algoritmu probíhají v tomto pořadí:

1. Přistoupení na první uzel ze spojového seznamu *toVisit*. Tento hexagon se nazývá kořen. Pokud je kořen bílý hexagon, uvažuje se v tomto algoritmu jako obarvený barvou strany (plánovaný krok - předpokládá se, že tento bílý hexagon bude obsazen).
2. Odebrání prvního uzlu se spojového seznamu *toVisit* a vložení kořene do kolekce *visited*.
3. Pokud je kořen bezpečně připojen ke straně desky *direction*, tak byla nalezena nejkratší potenciální cesta. V tom případě se smaže obsah spojového seznamu *toVisit*. Pomocí smyčky do-while se pak naplní spojový seznam *path* hexagony tvořící potenciální cestu k šestiúhelníku, ze kterého se SemiSBFS spouštěl. Smyčka do-while se dá popsat následujícím způsobem:
   - Pokud má kořen bílou barvu, přidá se na začátek spojového seznamu *path*.
   - Novým kořenem se stává kořen uložený v property Preceeding aktuálního kořene.
   - OPAKUJ DOKUD SE KOŘEN NEBUDE ROVNAT *null*.
4. Pokud kořen není bezpečně připojen ke straně desky *direction*, proběhnou tyto čtyři kroky:
   - Každý sousední hexagon kořene obarvený barvou strany se přidá na začátek spojového seznamu *toVisit*, pokud se tento hexagon nevyskytuje v kolekcích *visited* a *toVisit*. Property *Preceeding* tohoto barevného hexagonu se nastaví na kořen.
   - Každý hexagon obarvený barvou strany tvořící bezpečně spojený most s kořenem se přidá na začátek spojového seznamu *toVisit*, pokud se tento hexagon nevyskytuje v kolekcích *visited* a *toVisit*. Property *Preceeding* tohoto barevného hexagonu se nastaví na kořen.
   - Každý bílý hexagon, jehož označením by vznikl bezpečně spojený most s kořenem se přidá na konec spojového seznamu *toVisit*, pokud se tento hexagon nevyskytuje v kolekcích *visited* a *toVisit*. Property *Preceeding* tohoto bílého hexagonu se nastaví na kořen.

- Každý sousední bílý hexagon kořene se přidá na konec spojového seznamu *toVisit*, pokud se tento hexagon nevyskytuje v kolekcích *visited* a *toVisit*. Property *Preceeding* tohoto bílého hexagonu se nastaví na kořen.

5. Pokud není spojový seznam *toVisit* prázdný, algoritmus se sám znovu rekurzivně zavolá. V případě, že je spojový seznam *toVisit* prázdný, algoritmus končí.

4.3.5 Detekce struktur na desce

Jelikož jsou struktury přesně definované svým tvarem nebo charakteristickým umístěním na desce, vyhledávají se snadno. Např. vyhledávání mostu se provádí hledáním dvou stejně barevných hexagonů, které se nachází v určitém rozestupu.

Pokud je určitá struktura detekována, spočítá se její konektivita. Hráče zajímají pouze struktury s konektivitou 1, které vyžadují pozornost. Není potřeba zabývat se strukturami s jinou konektivitou (jsou neohrožené, nebo zablokované).

Stejný postup se aplikuje i na oponentovy struktury. Pokud některá z protivníkových struktur má konektivitu 1, hráč ji může ve svém tahu zablokovat.

4.3.6 Evaluace herní desky

Evaluace herní desky slouží k nalezení skupin na desce, hledání potenciálních cest a nejlepších kroků z těchto skupin ke stranám desky a vyhodnocení té nejlépe umístěné skupiny. Obecný algoritmus pro evaluaci desky již existuje a byl popsán v knize *Hex Strategy: Making the Right Connections* [4, s. 127]. Vlastní implementace hry operuje s novým experimenálním způsobem evaluace herního plánu, který byl tímto obecným algoritmem inspirován.

Vlastní implementace evaluace desky se dá rozdělit do několika bodů. Stejný postup se provede v jednom tahu pro oba hráče.

- Nalezení všech skupin na desce
- Kalkulace konektivit všech skupin hráče
- Identifikace nejsilnější skupiny

Následující body detailněji popisují jednotlivé kroky vlastního algoritmu evaluace desky.

1. **Nalezení všech skupin na desce**

   Obecně popsaný algoritmus evaluace desky operuje pouze s analýzou jednotlivých hexagonů a řetězů. Vlastní algoritmus pracuje s analýzou na úrovni skupin. Pro následnou práci s nimi je nutné vytvořit si množinu všech skupin hráče, které se nacházejí na herní desce.

2. **Kalkulace konektivit všech skupin hráče**

   Každá detekovaná skupina ve vlastním algoritmu je charakteristická několika konektivitami.

(a) Konektivita k první straně

(b) Konektivita ke druhé straně

(c) Celková konektivita skupiny

Konektivita ke straně je konektivita nejkratší potenciální cesty mezi některým kamenem ze skupiny a stranou. Celková konektivita skupiny je definována jako součet obou konektivit ke stranám. Tyto konektivity je nutno v tomto kroku algoritmu spočítat.

Tento algoritmus je univerzální pro oba hráče a proto nelze konkrétně definovat, ke které straně se konektivita od skupiny počítá. Pokud se jedná o skupinu modrého hráče, první strana představuje levý okraj a druhá strana představuje pravý okraj. V případě červeného hráče se první stranou rozumí spodní okraj a druhou stranou horní okraj desky.

Pro kalkulaci konektivit ke stranám a hledání potenciálních cest se používá smíšený průchod grafem (za předpokladu, že nejsou kameny skupiny bezpečně připojeny ke straně). Jeden průchod SemiSBFS nalezne vždy pouze jednu nejkratší potenciální cestu. Aby se úspěšně nalezly všechny různé nejkratší potenciální cesty, je nutné SemiSBFS spustit od každého hexagonu, který se nachází v množině bílých hexagonů dané skupiny. Spouštění SemiSBFS od každého bílého hexagonu skupiny znamená spouštění tohoto algoritmu od každého možného prvního bílého hexagonu z každé možné potenciální cesty.

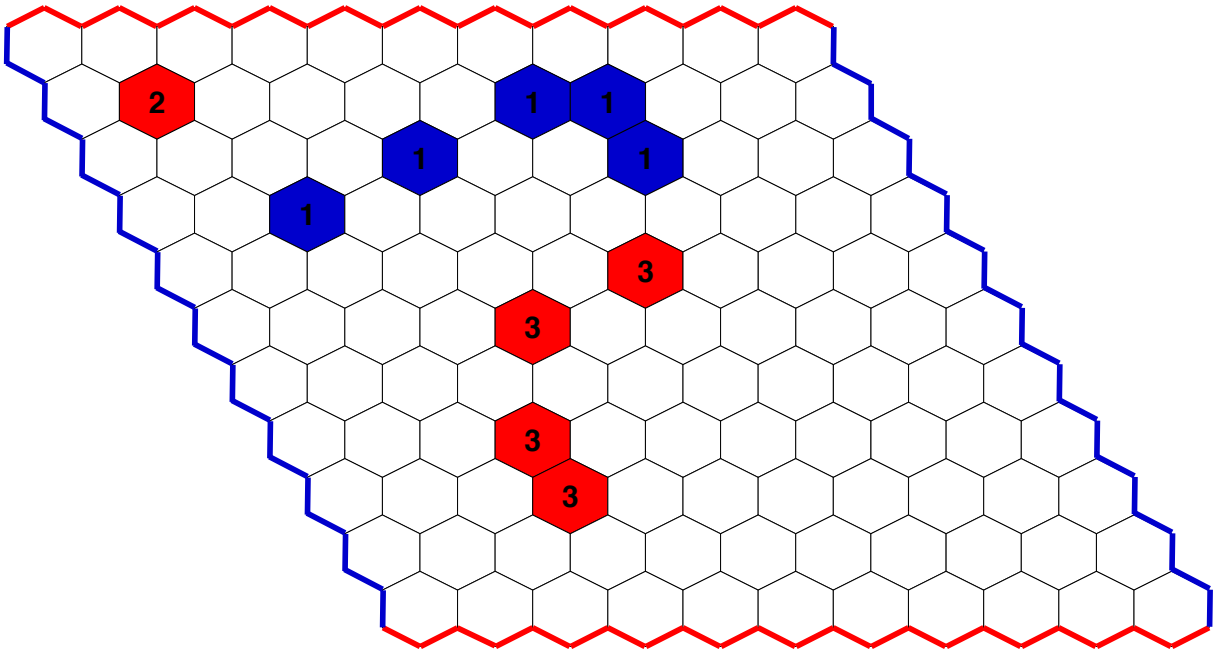
Když existuje více stejně dlouhých potenciálních cest, vybírá se nejvhodnější z nich. Pokud položením kamene na první bílý hexagon v této potenciální cestě vznikne bezpečně spojený most s kamenem nacházející se ve skupině, je tato cesta vyhodnocena jako nejvhodnější. Pokud existuje pouze jedna nejkratší potenciální cesta, stává se automaticky cestou nejvhodnější.

Bílý hexagon, od kterého se spouštěl smíšený průchod, jehož nejkratší potenciální cesta byla vyhodnocena jako nejvhodnější, je vyhodnocen jako nejlepší krok ze skupiny. Konektivity ke stranám a první bílé hexagony všech nejkratších potenciálních cest se uloží do analyzované skupiny.

3. **Identifikace nejsilnější skupiny**

   Nejsilnější skupina má ze všech skupin nejmenší celkovou konektivitou. Pokud existují dvě skupiny, které mají stejnou celkovou konektivitu, tak jsou tyto skupiny vyhodnoceny jako ekvivalentně silné. Jedna z těchto skupin se označí jako nejsilnější skupina hráče na desce. Z předchozího kroku algoritmu již známe konektivity a nejkratší potenciální cesty ke stranám desky.

Ukázka vlastní evaluace Tato podkapitola ilustruje postup u vlastní evaluace herní desky. Výchozí situaci popisuje obrázek č. 33. Kameny jsou označeny čísly, do které skupiny daný kámen patří.



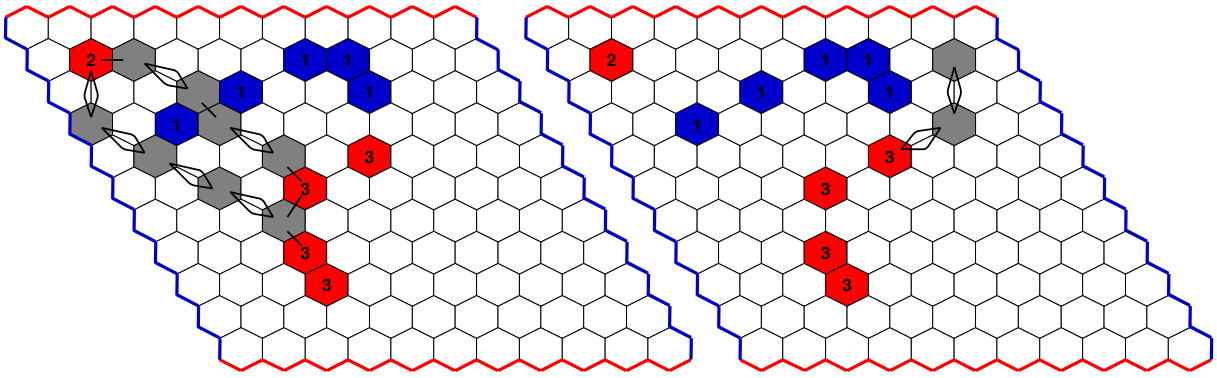
Obrázek 33: Herní deska k evaluaci

Evaluace červeného hráče Na herní desce se nachází celkem 2 skupiny červeného hráče. Položený kámen skupiny č. 2 je bezpečně připojen k horní straně desky pomocí šablony *II* a kameny druhé červené skupiny jsou bezpečně připojeny ke spodní straně pomocí šablon *IIIa, IIIb* a *IIIc*.

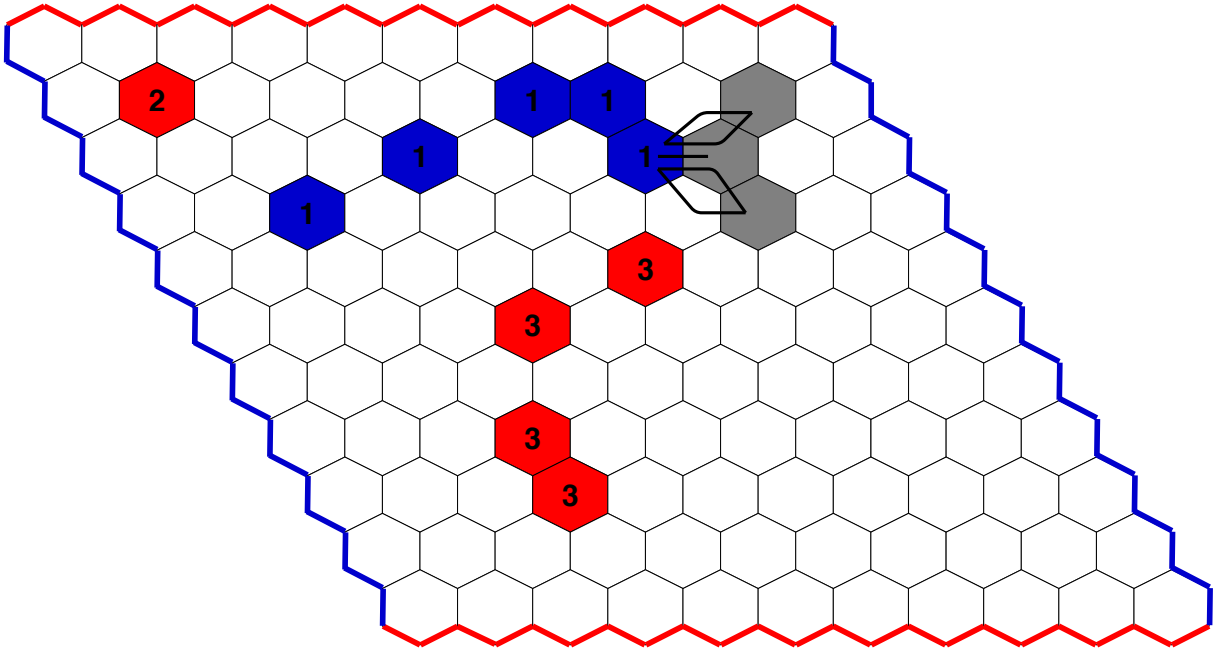
Po smíšených průchodem grafem známe konektivitu jednotlivých skupin. Konektivita skupiny č. 2 ke spodní straně desky je 4, zatímco konektivita skupiny č. 3 k hornímu okraji je 2. Jako nejsilnější skupina na desce byla tedy vyhodnocena skupina č. 3. Obrázek č. 34 ilustruje vybrané průchody po SemiSBFS. Šedé hexagony označují hexagony nejkratších potenciálních cest.

Ze skupiny č. 2 existuje více různých stejně dlouhých potenciálních cest. Pro přehlednost jsou ilustrovány pouze dvě. Jako nejvhodnější je zvolena cesta u modrého okraje desky, protože prvním krokem ze skupiny se vytvoří most od barevného hexagonu této skupiny k nově položenému kamenu. Kameny skupiny č. 3 jsou bezpečně připojeny ke spodní straně, není nutné přes ně zobrazovat průchody.

Evaluace modrého hráče Na herní desce se nachází jedna modrá skupina. Po SemiSBFS průchodech zjistíme, že existují celkem tři stejně dlouhé potenciální cesty, kterými může modrý hráč bezpečně připojit položené kameny ze své jediné a tudíž nejsilnější skupiny ke straně. Hexagony těchto cest jsou vyznačeny šedou barvou v obrázku č. 35. Pokud by byl na tahu červený hráč, všechny tyto cesty by byly zablokovatelné v jednom tahu, protože se všechny průchody překrývají v jednom hexagonu, viz kapitola 3.2.2.



Obrázek 34: Průchody a nejkratší cesty červeného hráče po evaluaci desky



Obrázek 35: Průchody a nejkratší cesty modrého hráče po evaluaci desky

Kroky algoritmu evaluace Algoritmus evaluace herní desky nemá žádné vstupní parametry. Po nalezení všech skupin hráče na desce se pro každou skupinu provádí následující postup:

1. Pokud nejsou barevné hexagony skupiny bezpečně připojeny k první straně, tak se od každého bílého hexagonu ve skupině spustí SemiSBFS k první straně. Postup po každém spuštění je následující:

   (a) Pokud je nalezená potenciální cesta nejkratší ze všech, stává se nejvhodnější cestou.

   (b) Pokud je nalezená potenciální cesta stejně dlouhá jako dříve nalezená nejkratší cesta, zkoumá se, jestli by obarvení prvního bílého hexagonu této cesty vytvořilo most z barevných hexagonů skupiny. Pokud ano, stává se tato cesta nejvhodnější potenciální cestou.

2. Pokud nejsou barevné hexagony skupiny bezpečně připojeny k druhé straně, tak se od každého bílého hexagonu ve skupině spustí SemiSBFS k druhé straně. Postup po každém spuštění je stejný jako po spouštění u první strany.

3. Nejvhodnější krok ze skupiny k první straně je vyhodnocen jako první hexagon z nejvhodnější potenciální cesty k této straně. Konektivita skupiny k první straně je určena počtem hexagonů v této cestě.

4. Nejvhodnější krok ze skupiny ke druhé straně je vyhodnocen jako první hexagon z nejvhodnější potenciální cesty k této straně. Konektivita skupiny ke druhé straně je určena počtem hexagonů v této cestě.

5. Spočítá se celková konektivita skupiny jako součet první a druhé konektivity. Pokud má tato skupina ze všech ostatních nejmenší celkovou konektivitu, stává se nejsilnější skupinou na desce.

4.3.7 Blokování soupeře

Blokování soupeře z dálky slouží jako způsob, jak odvrátit protivníkovo připojení ke straně. Samotný princip a ukázky tohoto postupu jsou vysvětleny v kapitole 3.2.1. Vlastní implementace je nastavena, aby začala blokovat nejefektivnějším způsobem až ve chvíli, kdy je to nezbytné. Tento případ nastane, když se protivníkův kámen přiblíží ke straně na vzdálenost 5 nebo 6 a tento oponentův kámen není k této straně bezpečně připojen. Protivníkovy kameny, které se nachází v menší vzdálenosti od strany budou s největší pravděpodobností k okraji bezpečně připojeny pomocí některé šablony.

Další způsob blokování ve vlastní implementaci nastane v případě, že konektivita k jedné straně nejsilnější protivníkovy skupiny se sníží na 1. V tomto případě proběhne hledání hexagonu, ve kterém se překrývají průchody nejkratších potenciálních cest, viz kapitola 3.2.2. Takový hexagon je vhodným kandidátem na blokování.

4.3.8 Vyhodnocení prvních tahů

První tahy bez swapovacího pravidla Vlastní implementace v prvním tahu červeného hráče vyhodnotí jako nejvhodnější pozici pro první kámen na desce o liché velikosti hexagon označený písmenem X, viz obrázek č. 36. V případě desky o sudé velikosti je nejlepší pozice pro první tah vyhodnocena jako středový hexagon.

Středový hexagon je na herní desce o liché velikosti pouze jeden (na obrázku č. 36 je označen písmenem S), na desce o sudé velikosti jsou středové hexagony dva - nachází se ve středu diagonály mezi tupými úhly herní desky. V případě položení kamene na středový hexagon desky o sudé velikosti se vybere náhodně jeden z nich.

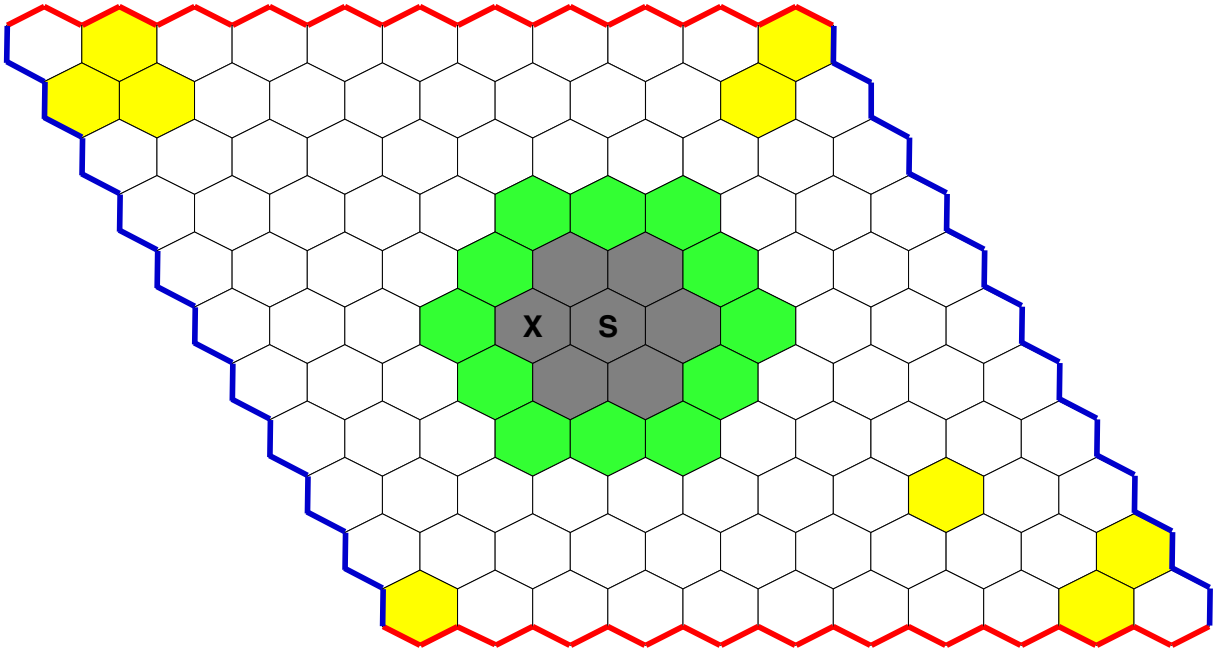
Pokud červený hráč položil kámen ve svém prvním tahu na jeden z hexagonů, který je vyznačený šedou barvou na obrázku č. 36, nejlepší první tah modrého hráče je vyhodnocen jako

blokování tohoto červeného kamene z dálky. Pokud je červený kámen položen na kteroukoli jinou pozici, modrý kámen se umístí na středový hexagon desky.

První tahy se swapovacím pravidlem Vlastní implementace pro první tah červeného hráče v tomto případě náhodně vybírá jeden z hexagonů, které jsou na obrázku č. 36 vyznačeny žlutě.

Pokud červený hráč umístil svůj kámen na jeden z hexagonů, který je na tomto obrázku vyznačen šedou nebo zelenou barvou, bude mu jeho kámen vyměněn za modrý. Pokud červený hráč umístil svůj kámen na kteroukoli jinou pozici, nejvhodnější pozice pro položení prvního modrého kamene je středový hexagon.

V případě výměny červeného kamene za modrý je pro následující tah červeného hráče vyhodnocen středový hexagon, pokud se modrý kámen nenachází na jednom z šedě označených hexagonů v obrázku č. 36. Pokud se na jednom z těchto hexagonů nachází modrý kámen, nejvhodnější tah je vyhodnocen jako blokování modrého kamene z dálky.



Obrázek 36: První tahy

4.3.9 Vyhodnocení ostatních tahů

Následující kapitola stručně popisuje kroky algoritmu, kterým se AI vlastní implementace řídí při vyhodnocení tahu. Tento postup není aplikovatelný pro první tahy, kdy hráči pokládají na desku své první kameny.

1. Algoritmus pro vyhodnocení příštího tahu začíná hledáním spojující cesty. Pokud spojující cesta existuje, AI zaplní bílý hexagon v této cestě, který je nutný pro vybudování řetězu

mezi stranami. Jestli byla nalezena oponentova spojující cesta, hráč s největší pravděpodobností prohrál a AI se pokusí marně blokovat protivníkovu spojující cestu.

2. Evaluace herní desky.

3. Na desce se poté naleznou zbytečné kruhy, jejichž prostřední hexagon byl obsazen protivníkem. Mosty v tomto zbytečném kruhu budou dále ignorovány.

4. Pokud po evaluaci desky má hráčova nejsilnější skupina celkovou konektivitu jedna, AI označí potřebný hexagon a vytvoří tak spojující cestu. Pokud nelze v jednom tahu vytvořit spojující cestu, ale protivníkova nejsilnější skupina má celkovou konektivitu jedna, proběhne hledání hexagonu, kterým se dá zablokovat vytvoření jeho spojující cesty. K tomuto hledání slouží zjištění překrývajících se průchodů k potenciálním cestám a detekce struktur s konektivitou 1, které jsou nezbytné pro vytvoření spojující cesty. Pokud takovýto hexagon existuje, znemožní se tím protivníkovi vytvoření spojující cesty.

5. Dále následuje detekce vlastních struktur na desce - nejprve mostů a pak šablon. Pokud má některá z těchto struktur konektivitu 1, algoritmus tuto strukturu zachrání.

6. Poté proběhne hledání vlastních žebříků. Žebřík bude budován, pouze pokud se z něj dá uniknout.

7. V dalším kroku budou ve stejném pořadí detekovány a zničeny protivníkovy mosty a šablony, které je možné zničit (mají konektivitu 1).

8. Dalším krokem v pořadí je detekce a zničení úniku z protivníkova žebříku, nebo případné provedení žebříkové vidličky.

9. V případě, že protivníkova nejsilnější skupina má menší celkovou konektivitu než vlastní nejsilnější skupina a je možno blokovat protivníka z dálky, provede se blokování.

10. Pokud se algoritmus dostane až do tohoto kroku, získává hráč momentum. Nejlepší tah je v tom případě vyhodnocen jako expanze - krok ze své nejsilnější skupiny. Vhodné kroky ze skupiny není nutné znovu hledat, nejkratší potenciální cesta k okraji a nejlépe ohodnocený hexagon pro krok zůstaly uloženy ve skupině po evaluaci herní desky. Tímto krokem hráč vytvoří, nebo se přiblíží k vytvoření spojovací cesty.

Je tedy zřejmé, že vlastní implementace používá algoritmus AI pracující primárně se strukturami, skupinami a vyhledáváním cest. Tento algoritmus je pro všechny tahy stejný a neměnný. Pořadí kroků, ve kterém se zkoumá umístění příštího kamene bylo takto vyhodnoceno a otestováno jako nejefektivnější. AI nepoužívá odhady dalších možných tahů protivníka ani vyhodnocení nejlepšího tahu pomocí výpočtů nebo bodového hodnocení hexagonů.

5 Přehled ostatních volně dostupných implementací

Tato kapitola obsahuje stručný přehled vybraných volně dostupných implementací hry Hex.

5.1 Hexy

Hexy je jedna z prvních softwarových implementací hry. Vývoj této implementace probíhal v letech 1999-2000. Hexy je určena pro systémy Microsoft Windows. Autorem Hexy je Vadim V. Anshelevich.

Algoritmy AI tohoto programu operují s tzv. *virtuálními cestami*. Virtuální cesta je označení pro bezpečné spojení dvou kamenů, které je ohodnoceno pomocí nejmenšího nutného počtu bílých hexagonů potřebných k udržení bezpečného spojení. Autor Hexy dále definuje algebru pro virtuální cesty. Tato algebra se používá pro vyhodnocení kombinací virtuálních cest za účelem vytvoření spojující cesty. [3, s. 4]

Hexy umožňuje nastavit celkem čtyři obtížnostní úrovně AI - začátečník, středně pokročilý, pokročilý a expert. Možné velikosti herní desky jsou 4 až 11.

5.2 Benzene

Benzene je kolekce open source knihoven a programů napsaných v C++ sloužící pro vývoj, testování a hraní hry Hex. Tato kolekce obsahuje mimo jiné dvě různé implementace AI - *MoHex* a *Wolve*, které umí hrát na deskách o velikosti 11, 13, 14 a 19. Za vývojem Benzene stojí katedra výpočetní vědy na univerzitě v Edmontonu. Na této katedře existuje od roku 1999 skupina akademických pracovníků věnující se výzkumu výpočetních technologií pro AI hry Hex, tzv. *University of Alberta Computer Hex Research Group*. Benzene bylo veřejně publikováno v roce 2013 a obsahuje implementace z roku 2011. [5] [6] [7]

5.2.1 MoHex

MoHex je nejkomplexnější implementace AI hry. Tato AI využívá hlavně tzv. *Monte Carlo* algoritmus. Jde o způsob zvolení nejvhodnějšího příštího tahu pomocí odehrání několika pseudonáhodných simulovaných partií po předpokládaném obsazení každého volného hexagonu. Odehrání pseudonáhodné partie se nazývá *playout*. Po skončení každé partie se uloží předpokládaný výsledek hry. Ohodnocení pozice daného hexagonu se počítá jako poměr výher z celkového počtu partií. Čím více pseudonáhodných partií je odehráno, tím je výsledek přesnější. Reálné výsledky tohoto algoritmu nikdy nejsou stoprocentně přesné, protože používá jistou míru náhody k předpokladu výsledku hry tím, že jí sám dohraje mnoha způsoby. I když je tento algoritmus náročný na výpočetní výkon, používá se pro AI na vysoké úrovni nejen pro deskové hry. [20]

MoHex mimo jiné zároveň používá pro vyhodnocení tahů také virtuální cesty jako Hexy. [7]

5.2.2 Wolve

Wolve je druhou implementací AI nacházející se v balíku Benzene. Původní označení této této implementace se jmenovalo *Mongoose*. Wolve používá tzv. *Alfa-beta ořezávání*. Alfa-beta ořezávání se používá pro vylepšení doby běhu algoritmu jménem *Minimax*, který slouží k nalezení nejvhodnějšího příštího tahu. Minimax ohodnotí podle určité bodovací funkce veškeré možné tahy a vybere z nich nejlépe ohodnocený. Způsob bodování tahu závisí na dané implementaci. [7] [19]

5.2.3 Solver

Za zmínku stojí také program Solver z balíku Benzene. Solver není hratelná implementace hry. Tento program slouží k nalezení všech možných variant hry na deskách různé velikosti. Všechny varianty hry na desce o rozměru 7 dokáže vyřešit za deset minut, varianty hry na desce o velikosti 8 přibližně za 30 hodin. Nalézt všechny možné varianty hry na desce o rozměru 9 trvalo Solveru dva roky. [7]

5.3 Six

Six je program pro systémy Linux/UNIX, na kterých běží desktopové prostředí KDE. Six byl také portován pro OS Microsoft Windows. Six byl vyvíjen v letech 2002 - 2006. Six podporuje hru na desce o velikosti 5 až 14. Autorem implementace je Gábor Melis.

Six je hratelný na čtyřech obtížnostních úrovních - začátečník, středně pokročilý, pokročilý a expert. Tato implementace používá podle svých webových stránek podobné algoritmy a virtuální cesty jako Hexy. [16]

5.4 HEX

HEX je volně dostupný program z roku 2010 napsaný v jazyce C# jako desktopová WPF aplikace. HEX nepodporuje Swap Rule. Tato implementace podporuje hru na desce o velikosti 5 až 12. Autorem programu je Anthony Steele.

HEX podporuje čtyři obtížnostní úrovně AI - Nízká, střední, dobrá a excelentní. AI této implementace používá vlastní algoritmus Minimax s Alfa-Beta ořezáváním. [21]

5.5 Hex: A Connection Game

Hex: A Connection Game je mobilní aplikace určená pro OS Android. Tato implementace dokáže hrát pouze na desce o velikosti 11. Autorem aplikace je studio Five Factorial.

Tato implementace podporuje čtyři obtížnostní úrovně - začátečník, středně pokročilý, pokročilý a expert. Podle popisu hry na Google Play používá při expertní obtížnosti tato implementace MoHex AI z balíku Benzene a pro zbylé úrovně vlastní AI bez dalšího popisu. [9]

6 Porovnání implemetací

Tato kapitola obsahuje porovnání vlastní implementace s vybranými implementacemi popsanými v předchozí kapitole.

6.1 The Computer Games Olympiad

The Computer Games Olympiad je událost, při které mezi sebou soutěží počítačové programy s umělou inteligencí ve hraní logických deskových her. První ročník této olympiády proběhl v roce 1989. Ne každý rok bývá olympiáda uspořádána. Poslední, 20. ročník olympiády proběhl v roce 2017. Jednou z disciplín je od roku 2000 také hra Hex. [10] [13]

6.1.1 Výsledky hry HEX na olympiádách

V následující kapitole jsou stručně uvedeny výsledky ostatních implementací z minulých herních olympiád. Některé implementace už nejsou dostupné, volně dostupné nebo z jiného důvodu dnes už nepoužitelné a proto nebyly popsány v kapitole 5. Tyto výsledky prezentují finální vítězné pořadí po vzájemných partiích implementací na herní desce o velikosti 11. Na každý ročník se musí implementace přihlašovat znovu a proto se některé implementace zúčastnily pouze vybraných ročníků olympiády. Na některých ročnících byly přihlášeny pouze dvě implementace a proto je třetí místo neobsazeno.

Tabulka č. 1 zobrazuje pořadí vítězných implementací hry Hex na herních olympiádách.

Rok	První místo	Druhé místo	Třetí místo
2000	Hexy	QueenBee	KillerBee
2003	Six	Mongoose	
2004	Six	Mongoose	
2006	Six	Mongoose	Hex Krieger
2008	Wolve	MoHex	Six
2009	MoHex	Wolve	Six
2010	MoHex	Wolve	MimHex
2011	MoHex	Wolve	Panoramex
2013	MoHex	Ezo	Jhinenox
2015	MoHex	DeepHex	Ezo
2017	MoHex	Ezo	Hexcited

Tabulka 1: Pořadí vítězných implementací hry Hex na herních olympiádách

V prvním ročníku olympiády, na kterém se objevila disciplína hry Hex, se stala Hexy vítěznou implementací. V dalších letech se stal vítězem program Six, který také používá virtuální cesty po vzoru Hexy. Až do roku 2008 se vždy na druhém místě umístila implementace s AI operující s algoritmem Minimax (resp. Alfa-Beta ořezáváním), dokud se v tomto roce na první

příčce neumístil Wolve. Od roku 2009 je Monte Carlo považován za nejlepší dostupný a dosud neporažený AI algoritmus. [7] [10] [13]

Počet vítězství použitých technik algoritmů AI z olympiád je následující:

6 Monte Carlo algoritmus

4 Virtuální cesty

1 Alfa-Beta ořezávání (Minimax)

Počet vítězství AI:

6 MoHex

3 Six

1 Wolve

1 Hexy

6.2 Porovnání vlastní implementace s ostatními implementacemi

HexGame byl porovnáván s programy Hexy, HEX, Hex: A Connection game a Six. Všechny partie byly odehrány na herní desce o velikosti 11, stejně jako na herní olympiádě. Herní deska o velikosti 11 je také jediná deska, se kterou jsou schopny pracovat všechny porovnávané implementace. Některé programy podporují nastavení obtížnostní úrovně AI. V tom případě byl program HexGame porovnán proti každé této obtížnostní úrovni.

Porovnávání probíhalo ručně - programy navzájem nijak nekomunikovaly. Každý tah musel být manuálně přenesen uživatelem do druhého programu. Následující tabulky s výsledky obsahují počty vítězství pro dané nastavení hry.

6.2.1 HexGame versus Hexy

Srovnání HexGame s Hexy probíhalo relativně rychle, protože doba pokládání kamenů Hexy se pohybuje v řádu sekund. I když má Hexy nastavení čtyř obtížnostních úrovní, provedené tahy této AI byly pro všechny tyto obtížnosti téměř totožné a velmi silné. Ať už se hrálo se swapovacím pravidlem nebo bez něj, HexGame nebyl schopný ani jednou Hexy porazit. I když byla často partie ze začátku hry vyrovnaná, ve všech případech nakonec vlastní program udělal špatný tah. Tímto jedním špatným tahem získala Hexy momentum a následně i vítězství.

Tabulky č. 2, 3, 4 a 5 zobrazují počty vítězství partií.

	Obtížnost AI Hexy			
Program	Začátečník	Středně pokročilý	Pokročilý	Expert
HexGame	0	0	0	0
Hexy	10	10	10	10

Tabulka 2: Výsledky partií proti Hexy, kdy první tah určoval HexGame bez swapovacího pravidla

	Obtížnost AI Hexy			
Program	Začátečník	Středně pokročilý	Pokročilý	Expert
HexGame	0	0	0	0
Hexy	10	10	10	10

Tabulka 3: Výsledky partií proti Hexy, kdy první tah určovala Hexy bez swapovacího pravidla

	Obtížnost AI Hexy			
Program	Začátečník	Středně pokročilý	Pokročilý	Expert
HexGame	0	0	0	0
Hexy	10	10	10	10

Tabulka 4: Výsledky partií proti Hexy, kdy první tah určoval HexGame se swapovacím pravidlem

	Obtížnost AI Hexy			
Program	Začátečník	Středně pokročilý	Pokročilý	Expert
HexGame	0	0	0	0
Hexy	10	10	10	10

Tabulka 5: Výsledky partií proti Hexy, kdy první tah určovala Hexy se swapovacím pravidlem

6.2.2 HexGame versus Six

Six využívá stejné principy virtuálních cest jako Hexy. I když Six také operuje se čtyřmi obtížnostními úrovněmi, tahy všech těchto úrovní byly velmi podobné a silné, stejně jako u partií s Hexy. HexGame nebyl schopný porazit Six ani v jedné partii ze stejných důvodů jako u Hexy. Následující tabulky č. 6, 7, 8 a 9 popisují počty vítězství partií.

	Obtížnost AI Six			
Program	Začátečník	Středně pokročilý	Pokročilý	Expert
HexGame	0	0	0	0
Six	10	10	10	10

Tabulka 6: Výsledky partií proti Six, kdy první tah určoval HexGame bez swapovacího pravidla

6.2.3 Vlastní implementace versus HEX

Testování proti programu HEX bylo časově náročné. Nejjednodušší obtížnost této AI pokládá kameny v řádu sekund, stejně jako Hexy. HEX jeden tah při střední obtížnosti běžně kalkuluje

	Obtížnost AI Six			
Program	Začátečník	Středně pokročilý	Pokročilý	Expert
HexGame	0	0	0	0
Six	10	10	10	10

Tabulka 7: Výsledky partií proti Six, kdy první tah určoval Six bez swapovacího pravidla

	Obtížnost AI Six			
Program	Začátečník	Středně pokročilý	Pokročilý	Expert
HexGame	0	0	0	0
Six	10	10	10	10

Tabulka 8: Výsledky partií proti Hexy, kdy první tah určoval HexGame se swapovacím pravidlem

	Obtížnost AI Six			
Program	Začátečník	Středně pokročilý	Pokročilý	Expert
HexGame	0	0	0	0
Six	10	10	10	10

Tabulka 9: Výsledky partií proti Six, kdy první tah určoval Six se swapovacím pravidlem

v řádu desítek sekund až minut. V případě dobré obtížnosti už kalkulace jednoho tahu probíhá v řádu desítek minut. Při použití excelentní obtížnosti trvá vyhodnocení řádově hodiny. Z tohoto důvodu nebyl HexGame porovnáván s touto excelentní obtížnostní úrovní tohoto programu.

HEX nepodporuje swapovací pravidlo. Všechny odehrané partie byly buď totožné, nebo velmi podobné. AI HEX operuje s vlastním algoritmem s Alfa-Beta ořezáváním. Je očividné, že vyhodnocovací funkce v tomto případě byla velmi složitě a s ohledem na výsledky a extrémní časovou náročnost neefektivně naimplementována.

HexGame dokázal porazit HEX ve všech případech testování. Celkem bylo proti HEX odehráno 50 partií. Oproti dobré obtížností úrovni bylo odehráno z důvodu časové náročnosti pokládání tahů méně partií. Detailní počty vítězství popisují tabulky č. 10 a 11.

	Obtížnost AI HEX			
Program	Nízká	Střední	Dobrá	Excelentní
HexGame	10	10	5	0
HEX	0	0	0	0

Tabulka 10: Výsledky partií proti HEX, kdy první tah určoval HEX

6.2.4 HexGame versus Hex: A Connection game

Porovnávání s mobilní aplikací *Hex: A Connection game* probíhalo relativně rychle, aplikace pokládá kameny téměř ihned po uživateli. Porovnáváním s touto aplikací bylo zároveň porovnáváno více implementací AI. Obtížnostní úrovně začátečník, středně pokročilý a pokročilý používají

	Obtížnost AI HEX			
Program	Nízká	Střední	Dobrá	Excelentní
HexGame	10	10	5	0
HEX	0	0	0	0

Tabulka 11: Výsledky partií proti HEX, kdy první tah určoval HexGame

nezdokumentovanou vlastní implementaci AI od vývojářského studia aplikace. Úroveň expert používá AI MoHex z balíku Benzene. Následující tabulky č. 12, 13, 14 a 15 popisují počty vítězství partí.

	Obtížnost AI			
Program	Začátečník	Středně pokročilý	Pokročilý	Expert (MoHex AI)
HexGame	10	10	10	0
Hex: A Connection game	0	0	0	10

Tabulka 12: Výsledky partií proti Hex: A Connection game, kdy první tah určoval HexGame bez swapovacího pravidla

	Obtížnost AI			
Program	Začátečník	Středně pokročilý	Pokročilý	Expert (MoHex AI)
HexGame	8	10	0	0
Hex: A Connection game	2	0	10	10

Tabulka 13: Výsledky partií proti Hex: A Connection game, kdy první tah určovala porovnávaná implementace bez swapovacího pravidla

	Obtížnost AI			
Program	Začátečník	Středně pokročilý	Pokročilý	Expert (MoHex AI)
HexGame	10	9	5	0
Hex: A Connection game	0	1	5	10

Tabulka 14: Výsledky partií proti Hex: A Connection game, kdy první tah určoval HexGame se swapovacím pravidlem

	Obtížnost AI			
Program	Začátečník	Středně pokročilý	Pokročilý	Expert (MoHex AI)
HexGame	10	8	10	0
Hex: A Connection game	0	2	0	10

Tabulka 15: Výsledky partií proti Hex: A Connection game, kdy první tah určovala porovnávaná implementace se swapovacím pravidlem

6.3 Zhodnocení testování a porovnání

V rámci testování bylo možno porovnat vlastní implementaci se čtyřmi programy a pěti různými implementacemi AI. Celkově bylo odehráno 530 partií, z toho celkem 150 dokázala vyhrát vlastní implementace. Celkový počet výher a proher každého porovnávaného programu zobrazuje tabulka č. 16.

Program	Výhry	Prohry
Hexy	160	0
Six	160	0
Hex: A Connection game	60	100
HEX	0	50

Tabulka 16: Počty vítězství a proher ostatních programů proti HexGame

Vlastní implementace nedokázala ani jednou porazit AI využívající virtuální cesty, nebo Monte Carlo algoritmus. Vlastní AI dosáhla porovnatelných výsledků s AI hry Hex: A Connection game. HexGame dokázal zvítězit ve všech případech nad programem HEX, jehož AI využívá vlastní Alfa-Beta ořezávání. Tabulka č. 17 zobrazuje počet výher a proher použitých technik algoritmů AI porovnávaných programů proti algoritmům AI vlastní implementace.

Technika algoritmů AI	Výhry	Prohry
Virtuální cesty	320	0
Monte Carlo algoritmus	40	0
Nezdokumentovaný algoritmus od studia Five Factorial	20	100
Minimax (Alfa-Beta ořezávání)	0	50

Tabulka 17: Počty vítězství a proher použitých technik algoritmů AI proti vlastní implementaci

Vítězné pořadí porovnávaných programů podle podle poměru výher a proher nad vlastním programem:

1. Hexy, Six
2. Hex: A Connection Game
3. HEX

Vítězné pořadí porovnávaných použitých technik algoritmů AI podle poměru výher a proher nad vlastní implementací algoritmů AI:

1. Monte Carlo algoritmus, virtuální cesty
2. Nezdokumentovaný algoritmus od studia Five Factorial
3. Minimax (Alfa-Beta ořezávání)

6.4 Hodnocení vlastní implementace

Vlastní program HexGame není špatný, protože dokázal porazit některé ostatní programy. Obtížnostní úroveň vlastní AI by se dala ohodnotit jako začátečnická až středně pokročilá. Vlastní AI je vhodná pro začínající a středně pokročilé hráče, ale není vhodná pro soutěžení na počítačové olympiádě. Všechny porovnávané implementace AI, které se zúčastnily olympiády jsou oproti vlastní AI mnohem silnější a propracovanější, protože byly vyvíjeny profesionály za použití pokročilých algoritmů v průběhu několika let. Obtížnostní úroveň vlastní AI se nachází na srovnatelné úrovni jako ostatní AI ostatních volně dostupných programů, které nebyly vyvíjeny profesionály za účelem soutěžení na olympiádě.

Algoritmy vlastní AI fungují na jiném principu oproti všem ostatním implementacím. Tyto vlastní algoritmy jsou funkční, ale po porovnání je zřejmé, že se nevyrovnají vyhodnocování tahů pomocí virtuálních cest, nebo pomocí algoritmu Monte Carlo. Potvrdilo se, že Monte Carlo je stále neporažený AI algoritmus, který je nejlepší pro implementaci herních AI.

V průběhu testování bylo očividné, že čím se herní deska více zaplňovala kameny, tím byl vlastní algoritmus více náchylný provést špatný tah. V kritických chvílích dokázala vlastní AI často vhodně zareagovat na nově vzniklou situaci. V mnoha případech se však stalo, že vlastní implementace reagovala na určitou hrozbu, kterou nebylo nutné odvrátit a tímto zbytečným tahem ztratila možnost vyhrát partii.

7 Závěr

V této práci byly objasněny pravidla, pojmy, principy a vybrané strategie hry Hex. Dále byly ukázány vybrané šestiúhelníkové struktury a jejich použití, které se hodí znát při analýze herní desky a promýšlení příštího tahu.

V rámci této práce byl také vytvořen program HexGame - vlastní implementace hry Hex s umělou inteligencí včetně testovacího prostředí. Architektura programu, jeho principy a logika vybraných algoritmů AI byly popsány v této práci. Je nutno podotknout, že se jedná o nový a zcela vlastní experimentální přístup umělé inteligence pro tuto hru, který se výrazně liší od všech použitých principů v ostatních programech. Efektivita této AI byla srovnávána s ostatními volně dostupnými implementacemi.

Vlastní program byl porovnáván s programy Hexy, Six, Hex: A Connection Game a HEX a pěti různými AI - Hexy, Six, HEX, MoHex a nezdokumentovanou AI vývojářského studia Five Factorial. Po odehrání celkem 530-ti partií a zhodnocení výsledků je očividné, že se nový přístup implementace umělé inteligence neosvědčil jako vhodný. Vlastní AI dokázala zvítězit přibližně ve 28% případů. Z výsledků testování vyplývá, že pro vytvoření nejefektivnější AI pro tuto hru je nutno použít algoritmus Monte Carlo, nebo techniku virtuálních cest.

Vlastní AI nebyla schopna porazit AI Hexy, Six a MoHex, které byly vítězné na herních počítačových olympiádách, ale dokázala zvítězit nad umělými inteligencemi HEX a Hex: A Connection Game, které nebyly vyvíjeny profesionály. Nově vytvořený program by se dal doporučit pro hru se začátečníky, nebo středně pokročilými hráči.

Literatura

[1] ANSHELEVICH, V. V. *Hexy Wins Hex Tournament.* The ICGA Journal, 23, (3), 2000, 181-184 [online]. [cit. 2018-04-11]. Dostupné z: http://vanshel.com/Hexy/Publications/Ansh-MSO-Results.pdf

[2] ANSHELEVICH, V. V. *The Game of Hex: The Hierarchical Approach.* [online]. [cit. 2018-04-11]. Dostupné z: http://vanshel.com/Hexy/Publications/VAnshelevich-MSRI-04.pdf

[3] ANSHELEVICH, V. V. *The Game of Hex: An Automatic Theorem Proving Approach to Game Programming.* [online]. [cit. 2018-04-21]. Dostupné z: http://vanshel.com/Hexy/Publications/VAnshelevich-01.pdf

[4] BROWNE, C. *Hex Strategy: Making the Right Connections.* A. K. Peters/CRC Press, 1. edice, 2000, ISBN 1568811179.

[5] CARTER, N. *Benzene @ SourceForge.* [online]. 2014 [cit. 2018-04-21]. Dostupné z: https://sourceforge.net/projects/benzene/

[6] Computer Hex research group. *Benzene.* [online]. 2011 [cit. 2018-04-21]. Dostupné z: http://benzene.sourceforge.net/

[7] Computer Hex research group. *Computer HEX.* University of Alberta, Department of computing science [online]. 2015 [cit. 2018-04-21]. Dostupné z: http://webdocs.cs.ualberta.ca/~hayward/hex/

[8] Česká televize. *AZ-kvíz.* [online]. [cit. 2018-04-15]. Dostupné z: http://www.ceskatelevize.cz/porady/1097147804-az-kviz/

[9] Five Factorial. *Hex: A Connection Game - Aplikace na Google Play.* [online]. 2017 [cit. 2018-04-25]. Dostupné z: https://play.google.com/store/apps/details?id=com.game.hex

[10] International Computer Games Association. *ICGA Tournament.* Tournaments between computer programs: chess, draughts, checkers, Go, backgammon, and more [online]. 2016 [cit. 2018-04-25]. Dostupné z: https://www.game-ai-forum.org/icga-tournaments/

[11] Klub přátel deskových her. *Hex.* [online]. [cit. 2018-04-11]. Dostupné z: http://www.deskovehry.info/pravidla/hex.htm

[12] KOVÁŘ, P. *Teorie grafů.* učební text [online]. 2016 [cit. 2018-04-10]. Dostupné z: http://homel.vsb.cz/~kov16/files/skriptum_teorie_grafu_rozsirene.pdf

[13] LEVY, D. *International Computer Games Association.* [online]. 2011 [cit. 2018-04-23]. Dostupné z: https://icga.org/

[14] MAARUP, T. *Everything You Ever Wanted to Know About Hex But Were Afraid to Ask*. UNIVERSITY OF SOUTHERN DENMARK, Department of Mathematics and Computer Science [online]. [cit. 2018-04-07]. Dostupné z: http://maarup.net/thomas/hex/hex3.pdf

[15] MARTIN, R. *HEX Wiki*. [online]. [cit. 2018-04-19]. Dostupné z: https://www.hexwiki.net/index.php

[16] MELIS, G. *Six - A Hex playing program for KDE*. [online]. [cit. 2018-04-21]. Dostupné z: http://quotenil.com/hex/six/

[17] NECKÁŘ, J. *Prohledávání do šířky*. Algoritmy.net [online]. 2016 [cit. 2018-04-23]. Dostupné z: https://www.algoritmy.net/article/1378/Prohledavani-do-hloubky

[18] NECKÁŘ, J. *Prohledávání do šířky*. Algoritmy.net [online]. 2016 [cit. 2018-04-23]. Dostupné z: https://www.algoritmy.net/article/1399/Prohledavani-do-sirky

[19] NĚMEC, J. *Šachové myšlení (2) - minimax*. [online]. 2006 [cit. 2018-04-25]. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=1141

[20] NIJSSEN, J. (Pim) A. M. and WINANDS, Mark H. M. *Playout Search for Monte-Carlo Tree Search in Multi-Player Games*. MAASTRICH UNIVERSITY, Faculty of Humanities and Sciences, Department of Knowledge Engineering [online]. Maastricht [cit. 2018-04-25]. Dostupné z: https://dke.maastrichtuniversity.nl/pim.nijssen/pub/bnaic12.pdf

[21] STEELE, A. *GitHub - AnthonySteele/Hex*. GitHub, Inc. [online]. 2010 [cit. 2018-04-21]. Dostupné z: https://github.com/AnthonySteele/Hex

[22] TRMPH. *HEX Wiki*. [online]. [cit. 2018-04-10]. Dostupné z: http://www.trmph.com/hexwiki/

A Příloha na CD

Přílohou této práce je CD s dokumentací a zdrojovými kódy vlastního programu. Toto CD obsahuje složku *HexGame*, ve které se nachází celý solution vlastní implementace. Vygenerovaná dokumentace kódu se nachází v umístění "HexGame\Documentation\Help\Documentation.chm". Celý zkompilovaný program (knihovna i spustitelný exe soubor) se nachází v adresáři "HexGame\GUI\bin\Release". Složka "HexGame\Diagrams" obsahuje ostatní dokumentaci - třídní diagram, závislostní diagram, Code Map celého programu pro Visual Studio a uživatelskou příručku programu.